Introdução ao R

Maurício Bueno

2022-06-26

Table of Contents

# Introdução

O R é uma linguagem e um ambiente dentro do qual se pode fazer uma infinidade de operações estatísticas e de processamento de dados. Foi concebido para ser colaborativo e gratuito. Qualquer pessoa pode criar uma função, ou um conjunto de funções (pacote) e disponibilizar a outros usuários… **gratuitamente**! Visite o o [site](https://cran.r-project.org/) do R e baixe o programa para instalação em seu computador.

Em associação ao R, costuma-se usar o RStudio (baixe e instale clicando [aqui](https://www.rstudio.com/)), que facilita o uso do R ao disponibilizar um conjunto de quatro janelas que funcionam para as seguintes funções:

1. digitação de texto (superior, à esquerda),
2. display de resultados (console) (inferior, à esquerda),
3. visualizador de objetos (arquivos) gerados ao realizar um trabalho (superior, à direita),
4. informações gerais: arquivos, plotagens, pacotes instalados, help, e visualizador de imagens (inferior, à direita).

Para a digitação dos comandos, deve-se abrir um novo Script. Faz-se isso com a seguinte linha de comando: File + New File + R Script, ou pressionando simultaneamente os botões CTRL+SHIFT+N. Fazendo isso, abrirá uma aba no quadrante superior esquerdo. Você pode abrir quantas abas desejar.

É possível rodar um comando no R de duas formas:

* Selecionando-o e clicando em “Run” na aba superior da área de digitação.
* Colocar o cursor em qualquer ponto da linha de comando e apertar CTRL+ENTER.

## Instalação e ativação de pacotes

Algumas funções não precisam da instalação de pacotes, pois já vêm com o R. São funções básicas (base), como operações matemáticas, plotagem, estatísticas básicas, entre outras. Outras funções estão em pacotes que precisam ser **instalados**, por meio do comando install.packages(), e **ativados**, por meio do comando library(). Então, vamos instalar alguns pacotes que iremos utilizar durante o curso.

# Instalação dos pacotes  
  
if(!require(readxl)) install.packages("readxl",repos = "http://cran.us.r-project.org")  
  
if(!require(tidyverse)) install.packages("tidyverse",repos = "http://cran.us.r-project.org")  
if(!require(knitr)) install.packages("knitr",repos = "http://cran.us.r-project.org")  
if(!require(kableExtra)) install.packages("kableExtra", repos = "http://cran.us.r-project.org")  
if(!require(expss)) install.packages("expss", repos = "http://cran.us.r-project.org")  
  
if(!require(tibble)) install.packages("tibble")  
# Ativacao dos pacotes ====  
  
library(readxl)  
library(tidyverse)  
library(knitr)  
library(expss)

Todo pacote do R tem uma documentação, com informações sobre as funções que o pacote contém. Uma forma de encontrar essa documentação é no site <rdocumentation.org>.

As funções contidas dentro dos pacotes ser acessadas digitando-se um ponto de interrogação (?) antes do nome da função. Por exemplo, se quisermos a documentação da função **hist** da base do R, é só rodar o comando:

A documentação aparecerá na aba **Help** do quadrante de visualizações de informações do RStudio.

## Operadores básicos

Existem também alguns operadores importantes no R, que você deve conhecer:  
Um deles é a seta para a esquerda, formada com o sinal de *menor* e o traço de *menos*: <-.

Atalho para esse operador: pressione simultaneamente as teclas ALT e -

Esse operador indica que o resultado de uma operação será salva em um **objeto**.

Por exemplo: o comando abaixo diz “*faça a soma de 3 com 4 e guarde o resultado num objeto chamado* ***soma****”*. Esse objeto (soma) aparecerá no environment (quadrante superior direito do RStudio).

soma <- 3 + 4

O comando “#” permite que você faça comentários, sem que o R o entenda como comando. Por exemplo:

soma <- 3 + 4 # Esse comando calcula a soma de 3 + 4 e salva no objeto "soma"

O texto que aparece após a #, em outra cor (verde, azul, etc.), é ignorado pelo R. Esses são os comandos mais básicos. Há muitos outros atalhos…

| atalho | o.que.faz |
| --- | --- |
| CTRL + ENTER | Executa a linha selecionada |
| CTRL + SHIFT + C | Comenta e descomenta a linha |
| CTRL + 1 | Passa o cursor para o script |
| CTRL + 2 | Passa o cursor para o console |
| SETA PARA CIMA (no console) | acessa o histórico de comandos anteriores |
| CTRL + ALT + SETA PARA ESQUERDA OU DIREITA | Navega entre as abas de script abertas |
| CTRL + SHIFT + ENTER | Executa o Script inteiro |
| CTRL + S | Salva o Script |
| CTRL + L | Limpa o console |
| ALT + SHIFT + K | Ver a lista de atalhos |

## Definir diretório de trabalho

Para facilitar o trabalho com salvamento e recuperação de arquivos, é recomendável que se defina um diretório de trabalho, onde todos os arquivos referentes à análise de dados serão salvos.

# para verificar o diretório atual, execute o comando:   
getwd()

## [1] "C:/Users/jmhbu/OneDrive/Documentos/R/R\_basico/Intro\_R"

# OBS.: se for necessário, navegue até o local desejado utilizando a aba "files"  
  
# defina sua pasta de trabalho utilizando a função:   
setwd("C:/Users/jmhbu/OneDrive/Documentos/R/R\_basico/Intro\_R")

## Principais operadores

O R pode ser usado como calculadora…

A tabela a seguir apresenta os principais operadores que usamos nos códigos em R.

| Operador | Descrição |
| --- | --- |
| + | Adição |
| - | Subtração |
| \* | Multiplicação |
| / | Divisão |
| : | Sequência |
| ^ | Exponencial |
| sqrt | Raiz |
| == | Igualdade |
| > | Maior que |
| < | Menor que |
| <= | Menor ou igual |
| >= | Maior ou igual |
| ! | Não |
| & | E |
| | | Ou |

Exercitando…

# operações matemáticas  
  
3+4

## [1] 7

5-2

## [1] 3

4\*2

## [1] 8

9/3

## [1] 3

sqrt(9)

## [1] 3

2^3

## [1] 8

1:10

## [1] 1 2 3 4 5 6 7 8 9 10

10:1

## [1] 10 9 8 7 6 5 4 3 2 1

#3 = 3  
3 == 3

## [1] TRUE

3 > 2

## [1] TRUE

3 < 2

## [1] FALSE

3 >= 2

## [1] TRUE

3 >= 3

## [1] TRUE

3 <= 2:5

## [1] FALSE TRUE TRUE TRUE

3 + 4 >= 14/2

## [1] TRUE

!3 == 5

## [1] TRUE

1 < 2 & 2 < 3

## [1] TRUE

1 == 2 | 2 > 3

## [1] FALSE

## Operações lógicas

Algumas palavras são reservadas no R para operações lógicas. As principais são TRUE, FALSE e NA.

| Operadores.lógicos | Significados |
| --- | --- |
| NA | Not available - dado faltante/indiponível |
| NaN | Not a number - indefinições matemáticas como 0/0, log(-1) |
| Inf | Infinito - número muito grande, como 1/0 e 10^310 |
| NULL | Representa ausência de objeto |
| TRUE | Condição é verdadeira |
| FALSE | Condição é falsa |

## Criando objetos

Os objetos que podem ser criados no R são:

* listas
* vetores
* matrizes
* array
* dataframes

Vamos definir os que mais usamos em nossas análises.

**Lista**: é uma coleção de informações.

minha\_lista <- list("s01","masculino",23,"ensino médio")  
class(minha\_lista)

## [1] "list"

# minha\_lista é um objeto do tipo lista, que guarda informações de diferentes tipos: id, sexo, idade, escolaridade  
# essas informações foram salvas num objeto do tipo lista, que pode ser visto no environment  
# informações não-numéricas aparecem entre "aspas" para o R reconhecer cada elemento dentro do objeto  
# as informações dessa lista poderiam ser de um sujeito num df, cada informação é de uma natureza  
# o comando "c", significa "concatenar". É usado para inserir mais de uma informação.

**Vetor**: é uma coleção de informações ou elementos **da mesma natureza**. Por exemplo, as variáveis que costumamos colocar nas colunas dos nossos bancos de dados são **vetores**.

meu\_vetor <- c("masculino","feminino","feminino","masculino","feminino")  
is.list(meu\_vetor)

## [1] FALSE

is.vector(meu\_vetor)

## [1] TRUE

is.data.frame(meu\_vetor)

## [1] FALSE

# meu\_vetor é um objeto do tipo vetor porque contem informações da mesma natureza. Poderiam ser informações sobre o sexo dos sujeitos em um banco de dados  
  
vetor1 <- c(1, 5, 3, -10)  
vetor2 <- c("a", "b", "c")  
  
class(vetor1)

## [1] "numeric"

class(vetor2)

## [1] "character"

# Se tentarmos misturar duas classes, o R vai apresentar um comportamento conhecido como coerção. Ele vai impor uma das classes aos objetos. Por exemplo:  
  
vetor <- c(1, 2, "a")  
  
vetor

## [1] "1" "2" "a"

class(vetor)

## [1] "character"

**Dataframe**: é um conjunto de vetores. Nossos bancos de dados, no R, recebem o nome de dataframes.

# criar vetor sexo, em que 1 é masculino e 2 é feminino.  
sexo <- c(1,2,1,1,2)  
is.vector(sexo)

## [1] TRUE

# pode-se criar um vetor semelhante, mas com letras  
# quando uma informação textual for inserida num objeto, ela tem que ir entre aspas.  
sexo\_cod <- c("m","f","m","m","f") # vetor com informações sobre sexo  
  
# uma alternativa a isso é transformar a variável sexo em uma variável fator  
  
sexo <- factor(sexo, levels = c(1,2), labels = c("masculino", "feminino"))  
  
  
idade <- c(25,32,78,12,NA) # vetor com informações de idade  
  
# A partir dos dois vetores anteriores é possível criar um dataframe (conjunto de vetores)  
df <- data.frame(sexo\_cod,idade)  
  
# acrescentando a variável escolaridade  
escolaridade <- c("superior","medio","fundamental","fundamental","medio")  
  
cbind(df,escolaridade) # cbind() é uma função da base que anexa colunas.

## sexo\_cod idade escolaridade  
## 1 m 25 superior  
## 2 f 32 medio  
## 3 m 78 fundamental  
## 4 m 12 fundamental  
## 5 f NA medio

df <- cbind(df,escolaridade)   
  
# acrescentando a variável id  
id <- 1:5  
df <- cbind(id,df)  
  
# Se quiser inserir um "s" antes do número de cada sujeito, podemos usar a função paste(), que concatena duas partes da informação que constar em uma coluna. Essa função diz: concatene um s com valores de 1 a 5, separados por nada.  
  
paste("s",1:5,sep="\_")

## [1] "s\_1" "s\_2" "s\_3" "s\_4" "s\_5"

id <- paste("s",1:5,sep="")  
  
# uma forma de salvar os resultados em uma nova variável diretamente no banco de dados é usando a função $ (cifrão). Quando digitamos o $ em seguida ao nome do objeto (dataframe), o R abre, automaticamente, abre uma janela de opções com as variáveis que existem no dataframe. Ao selecionar "id", por exemplo, os novos dados serão salvos SOBRE os dados existentes. Caso se queira CRIAR uma nova variável, basta digitar o nome de uma variável inexistente no dataframe.  
  
df$id <- paste("s",1:5,sep="")  
  
# se quiser inserir um novo sujeito no dataframe  
  
suj6 <- list(id="s6",sexo\_cod="m",idade=34,escolaridade="superior")  
suj6 <- list("s6","m",34,"superior")  
  
  
rbind(df,suj6)

## id sexo\_cod idade escolaridade  
## 1 s1 m 25 superior  
## 2 s2 f 32 medio  
## 3 s3 m 78 fundamental  
## 4 s4 m 12 fundamental  
## 5 s5 f NA medio  
## 6 s6 m 34 superior

df <- rbind(df,suj6)  
glimpse(df)

## Rows: 6  
## Columns: 4  
## $ id <chr> "s1", "s2", "s3", "s4", "s5", "s6"  
## $ sexo\_cod <chr> "m", "f", "m", "m", "f", "m"  
## $ idade <dbl> 25, 32, 78, 12, NA, 34  
## $ escolaridade <chr> "superior", "medio", "fundamental", "fundamental", "medio~

# inserir a variável renda  
# seq(from,to,by,lenth)  
  
df$renda <- seq(1000, by=500,length=6)  
  
# ou  
#renda <- seq(1000, by=500,length=6)  
#df <- cbind(df,renda)  
  
# inserir uma variável "bonus", referente a um abono de 500 reais a cada sujeito.  
# usamos a função rep(), que é semelhante à variável seq(), sem o argumento "by", porque a informação é sempre a mesma  
df$bonus <- rep(500,6)  
# ou  
#bonus <- rep(500,6)  
#df <- cbind(df,bonus)  
  
# calcular renda\_total, que é a soma das variáveis renda e bonus  
df$renda\_total <- df$renda + df$bonus  
  
# inserir a variável "Estado Civil"  
df$"Estado Civil" <- c("casado/a","solteiro/a","viúvo/a","solteiro/a","casado/a","casado/a")

# Funções para visualização de dados

df

## id sexo\_cod idade escolaridade renda bonus renda\_total Estado Civil  
## 1 s1 m 25 superior 1000 500 1500 casado/a  
## 2 s2 f 32 medio 1500 500 2000 solteiro/a  
## 3 s3 m 78 fundamental 2000 500 2500 viúvo/a  
## 4 s4 m 12 fundamental 2500 500 3000 solteiro/a  
## 5 s5 f NA medio 3000 500 3500 casado/a  
## 6 s6 m 34 superior 3500 500 4000 casado/a

names(df)

## [1] "id" "sexo\_cod" "idade" "escolaridade" "renda"   
## [6] "bonus" "renda\_total" "Estado Civil"

str(df)

## 'data.frame': 6 obs. of 8 variables:  
## $ id : chr "s1" "s2" "s3" "s4" ...  
## $ sexo\_cod : chr "m" "f" "m" "m" ...  
## $ idade : num 25 32 78 12 NA 34  
## $ escolaridade: chr "superior" "medio" "fundamental" "fundamental" ...  
## $ renda : num 1000 1500 2000 2500 3000 3500  
## $ bonus : num 500 500 500 500 500 500  
## $ renda\_total : num 1500 2000 2500 3000 3500 4000  
## $ Estado Civil: chr "casado/a" "solteiro/a" "viúvo/a" "solteiro/a" ...

glimpse(df)

## Rows: 6  
## Columns: 8  
## $ id <chr> "s1", "s2", "s3", "s4", "s5", "s6"  
## $ sexo\_cod <chr> "m", "f", "m", "m", "f", "m"  
## $ idade <dbl> 25, 32, 78, 12, NA, 34  
## $ escolaridade <chr> "superior", "medio", "fundamental", "fundamental", "med~  
## $ renda <dbl> 1000, 1500, 2000, 2500, 3000, 3500  
## $ bonus <dbl> 500, 500, 500, 500, 500, 500  
## $ renda\_total <dbl> 1500, 2000, 2500, 3000, 3500, 4000  
## $ `Estado Civil` <chr> "casado/a", "solteiro/a", "viúvo/a", "solteiro/a", "cas~

head(df)

## id sexo\_cod idade escolaridade renda bonus renda\_total Estado Civil  
## 1 s1 m 25 superior 1000 500 1500 casado/a  
## 2 s2 f 32 medio 1500 500 2000 solteiro/a  
## 3 s3 m 78 fundamental 2000 500 2500 viúvo/a  
## 4 s4 m 12 fundamental 2500 500 3000 solteiro/a  
## 5 s5 f NA medio 3000 500 3500 casado/a  
## 6 s6 m 34 superior 3500 500 4000 casado/a

ncol(df)

## [1] 8

nrow(df)

## [1] 6

view(df)  
unique(df$"Estado Civil")

## [1] "casado/a" "solteiro/a" "viúvo/a"

unique(df$idade)

## [1] 25 32 78 12 NA 34

df\_copia <- df  
df <- df\_copia

# Funções para manipulação de dados

Muitas vezes, precisamos manipular os dados de um dataframe, seja para renomear colunas, selecionar linhas ou colunas, adicionar variáveis ou sujeitos, entre outras. Vejamos algumas delas.

## Alteração do nome de variáveis

# Alterar o nome da variável "sexo\_cod"  
df <- rename(df,sexo = sexo\_cod)  
  
# Alterar o nome da variável "Estado Civil"  
df <- rename(df,estado\_civil = "Estado Civil")

## Selecionar linhas e colunas

# Selecionar linhas e colunas: dataframe[linhas,colunas]  
  
df[1:3, ] # seleção das três primeiras linhas

## id sexo idade escolaridade renda bonus renda\_total estado\_civil  
## 1 s1 m 25 superior 1000 500 1500 casado/a  
## 2 s2 f 32 medio 1500 500 2000 solteiro/a  
## 3 s3 m 78 fundamental 2000 500 2500 viúvo/a

df[ ,1:3] # seleção das três primeiras colunas

## id sexo idade  
## 1 s1 m 25  
## 2 s2 f 32  
## 3 s3 m 78  
## 4 s4 m 12  
## 5 s5 f NA  
## 6 s6 m 34

df[1:3,1:3] # seleção das três primeiras linhas e das três primeiras colunas

## id sexo idade  
## 1 s1 m 25  
## 2 s2 f 32  
## 3 s3 m 78

# Se que quisermos selecionar as linhas 1, 3 e 6, como fazemos?  
df[c(1,3,5), ]

## id sexo idade escolaridade renda bonus renda\_total estado\_civil  
## 1 s1 m 25 superior 1000 500 1500 casado/a  
## 3 s3 m 78 fundamental 2000 500 2500 viúvo/a  
## 5 s5 f NA medio 3000 500 3500 casado/a

# se quisermos selecionar as colunas id, idade e renda\_total  
df[ ,c(1,3,7)]

## id idade renda\_total  
## 1 s1 25 1500  
## 2 s2 32 2000  
## 3 s3 78 2500  
## 4 s4 12 3000  
## 5 s5 NA 3500  
## 6 s6 34 4000

df[ ,c("id","idade","renda\_total")]

## id idade renda\_total  
## 1 s1 25 1500  
## 2 s2 32 2000  
## 3 s3 78 2500  
## 4 s4 12 3000  
## 5 s5 NA 3500  
## 6 s6 34 4000

df[ ,c("renda\_total","idade","id")]

## renda\_total idade id  
## 1 1500 25 s1  
## 2 2000 32 s2  
## 3 2500 78 s3  
## 4 3000 12 s4  
## 5 3500 NA s5  
## 6 4000 34 s6

## Filtrar linhas inteiras

# Selecionar somente as linhas dos participantes do sexo masculino  
df[df$sexo == "m", ]

## id sexo idade escolaridade renda bonus renda\_total estado\_civil  
## 1 s1 m 25 superior 1000 500 1500 casado/a  
## 3 s3 m 78 fundamental 2000 500 2500 viúvo/a  
## 4 s4 m 12 fundamental 2500 500 3000 solteiro/a  
## 6 s6 m 34 superior 3500 500 4000 casado/a

# Selecionar somente as linhas dos participantes do sexo masculino com renda\_total maior ou igual a 2500 reais.  
df[df$sexo == "m" & df$renda\_total >= 2500, ]

## id sexo idade escolaridade renda bonus renda\_total estado\_civil  
## 3 s3 m 78 fundamental 2000 500 2500 viúvo/a  
## 4 s4 m 12 fundamental 2500 500 3000 solteiro/a  
## 6 s6 m 34 superior 3500 500 4000 casado/a

## Alterar o conteúdo dentro de uma variável

# Alterar o conteúdo dentro de uma variável  
  
df$sexo[df$sexo == "f"] <- 0  
df$sexo[df$sexo == "m"] <- 1

## Alterar o formato de variáveis

# visualisar que a variável sexo está como character, mas também foi possível perceber que outras variáveis também não estão no formato ideal. Vamos acertar todas elas.  
glimpse(df)

## Rows: 6  
## Columns: 8  
## $ id <chr> "s1", "s2", "s3", "s4", "s5", "s6"  
## $ sexo <chr> "1", "0", "1", "1", "0", "1"  
## $ idade <dbl> 25, 32, 78, 12, NA, 34  
## $ escolaridade <chr> "superior", "medio", "fundamental", "fundamental", "medio~  
## $ renda <dbl> 1000, 1500, 2000, 2500, 3000, 3500  
## $ bonus <dbl> 500, 500, 500, 500, 500, 500  
## $ renda\_total <dbl> 1500, 2000, 2500, 3000, 3500, 4000  
## $ estado\_civil <chr> "casado/a", "solteiro/a", "viúvo/a", "solteiro/a", "casad~

str(df)

## 'data.frame': 6 obs. of 8 variables:  
## $ id : chr "s1" "s2" "s3" "s4" ...  
## $ sexo : chr "1" "0" "1" "1" ...  
## $ idade : num 25 32 78 12 NA 34  
## $ escolaridade: chr "superior" "medio" "fundamental" "fundamental" ...  
## $ renda : num 1000 1500 2000 2500 3000 3500  
## $ bonus : num 500 500 500 500 500 500  
## $ renda\_total : num 1500 2000 2500 3000 3500 4000  
## $ estado\_civil: chr "casado/a" "solteiro/a" "viúvo/a" "solteiro/a" ...

# mudar variável sexo para factor (note que está como caractere)  
df$sexo <- as.numeric(df$sexo)  
df$sexo <- as.factor(df$sexo) # factor é uma variável nominal  
# df$sexo <- c("m","f","m","m","f","m")  
  
# df$idade <- as.numeric(df$idade)  
df$escolaridade <- as.factor(df$escolaridade)

## Atribuir níveis às variáveis de fator

# atribuir levels aos valores 0 e 1 da variável sexo  
levels(df$sexo) <- c("feminino","masculino")  
levels(df$escolaridade) <- c("fundamental","médio","superior")

## Alterar a ordem das variáveis no dataframe

# Alterar a ordem das variáveis no banco de dados  
df <- df[ ,c("id","sexo","idade","escolaridade","estado\_civil",  
 "renda","bonus","renda\_total")]

## Combinar dataframes

Nesse caso, temos o dataframe variables, com a informação de cinco variáveis dos mesmos sujeitos que constam no dataframe df. Então, gostaríamos de juntar esses dois dataframes. Para isso podemos usar duas funções.

# Juntar a planilha importada no dataframe df  
df <- cbind(df,variables)

## Atribuição de labels

# Atribuir labels às variáveis v1 a v5  
# vamos supor que essa variávels correspondessem às seguintes perguntas:  
# v1 = Você costuma se atrasar para as aulas?  
# v2 = Quanto você confia na sua capacidade de aprender?  
# v3 = Quão bem você se relaciona com outras pessoas na sua escola?  
# v4 = Quanto você gosta de sua escola?  
# v5 = Quanto você procrastina para estudar?  
# para isso vamos usar a função apply\_labels do pacote expss  
  
df <- apply\_labels(df,  
 v1 = "Você costuma se atrasar para as aulas?",  
 v2 = "Quanto você confia na sua capacidade de aprender?",  
 v3 = 'Quão bem você se relaciona com outras pessoas na sua escola?',  
 v4 = 'Quanto você gosta de sua escola?',  
 v5 = 'Quanto você procrastina para estudar?')  
  
str(df)

## 'data.frame': 6 obs. of 13 variables:  
## $ id : chr "s1" "s2" "s3" "s4" ...  
## $ sexo : Factor w/ 2 levels "feminino","masculino": 2 1 2 2 1 2  
## $ idade : num 25 32 78 12 NA 34  
## $ escolaridade: Factor w/ 3 levels "fundamental",..: 3 2 1 1 2 3  
## $ estado\_civil: chr "casado/a" "solteiro/a" "viúvo/a" "solteiro/a" ...  
## $ renda : num 1000 1500 2000 2500 3000 3500  
## $ bonus : num 500 500 500 500 500 500  
## $ renda\_total : num 1500 2000 2500 3000 3500 4000  
## $ v1 :Class 'labelled' num 2 4 3 5 1 2  
## .. .. LABEL: Você costuma se atrasar para as aulas?   
## $ v2 :Class 'labelled' num 4 5 3 5 2 1  
## .. .. LABEL: Quanto você confia na sua capacidade de aprender?   
## $ v3 :Class 'labelled' num 1 3 2 4 2 3  
## .. .. LABEL: Quão bem você se relaciona com outras pessoas na sua escola?   
## $ v4 :Class 'labelled' num 5 1 3 2 4 1  
## .. .. LABEL: Quanto você gosta de sua escola?   
## $ v5 :Class 'labelled' num 3 2 4 1 5 3  
## .. .. LABEL: Quanto você procrastina para estudar?

glimpse(df)

## Rows: 6  
## Columns: 13  
## $ id <chr> "s1", "s2", "s3", "s4", "s5", "s6"  
## $ sexo <fct> masculino, feminino, masculino, masculino, feminino, masc~  
## $ idade <dbl> 25, 32, 78, 12, NA, 34  
## $ escolaridade <fct> superior, médio, fundamental, fundamental, médio, superior  
## $ estado\_civil <chr> "casado/a", "solteiro/a", "viúvo/a", "solteiro/a", "casad~  
## $ renda <dbl> 1000, 1500, 2000, 2500, 3000, 3500  
## $ bonus <dbl> 500, 500, 500, 500, 500, 500  
## $ renda\_total <dbl> 1500, 2000, 2500, 3000, 3500, 4000  
## $ v1 <labelled> 2, 4, 3, 5, 1, 2  
## $ v2 <labelled> 4, 5, 3, 5, 2, 1  
## $ v3 <labelled> 1, 3, 2, 4, 2, 3  
## $ v4 <labelled> 5, 1, 3, 2, 4, 1  
## $ v5 <labelled> 3, 2, 4, 1, 5, 3

view(df)

# Função ifelse()

A função ifelse() é bastante útil para a manipulação de dados. Ela deve ser escrita da seguinte forma: ifelse("condição", "se sim", "se não") Traduzindo: observe a condição… se ela ocorrer, faça isso… se ela não ocorrer, faça aquilo. Exemplos:

# se quisermos classificar a renda em dois grupos: menor igual a R$2000 e maior que R$2000, podemos usar a função ifelse()  
# ifelse(condição,se sim, se não)  
# traduzindo....  
# observe se, na variável renda, o valor é menor ou igual a 2000  
# se for, então atribua 1  
# se não for, então atribua 2.  
  
ifelse(df$renda <= 2000,1,2)

## [1] 1 1 1 2 2 2

# Se quisermos salvar esses dados como uma coluna do df...  
df$renda\_cod <- ifelse(df$renda <= 2000,1,2)  
  
# se quisermos criar uma coluna atribuindo 1 para homens e 0 para os demais.  
ifelse(df$sexo == "masculino",1,0)

## [1] 1 0 1 1 0 1

# se quisermos identificar homens com escolaridade fundamental  
ifelse(df$sexo == "masculino" & df$escolaridade == "fundamental",1,0)

## [1] 0 0 1 1 0 0

# se quisermos identificar homens com escolaridade fundamental OU pessoa casadas.  
ifelse(df$sexo == "masculino" & df$escolaridade == "fundamental" | df$estado\_civil == "casado/a",1,0)

## [1] 1 0 1 1 1 1

# se quisermos criar uma nova variável atribuindo 1 para solteiro, 2 para casado e 3 para viúvo... tínhamos visto o formato  
  
#df$escolaridade[df$escolaridade == "fundamental"] <- 1  
#df$escolaridade[df$escolaridade == "médio"] <- 2  
#df$escolaridade[df$escolaridade == "superior"] <- 3  
  
df$escola\_cod <- ifelse(df$escolaridade == "fundamental",1,  
 ifelse(df$escolaridade == "médio",2,3))  
  
ifelse(df$v1 == 1,5,  
 ifelse(df$v1 == 2,4,  
 ifelse(df$v1 == 3,3,  
 ifelse(df$v1 == 4,2,1))))

## [1] 4 2 3 1 5 4

# Também é possível recodificar variáveis de jeito mais fácil, usando uma função do pacote expss.  
# Note que há questões no sentido de uma boa adaptação aos estudos e frases no sentido contrário. Nesses casos, não podemos somar os itens diretamente porque um valor 5 em v1 significa má adaptação, enquanto o mesmo valor 5 significa boa adaptação em v2. Então, não podemos somar coisas que têm significados diferentes.  
# Para lidar com isso, geralmente, fazemos a inversão de itens que estão na direção contrária do que queremos avaliar. Por exemplo, se queremos avaliar "adaptação escolar", então invertemos os itens v1 e v5, que estão na direção contrária.  
# Inverter significa transformar 1 em 5, 2 em 4, 3 em 3, 4 em 2 e 5 em 1.   
# Para isso, usamos a função recode() do expss.  
  
recode(df[ ,c("v1","v2","v3","v4","v5")]) <- c(1~5,2~4,3~3,4~2,5~1)  
recode(df[ ,c("v1","v5")]) <- c(1~5,2~4,3~3,4~2,5~1)

# Pacote tidyverse

O tidyverse é uma família de pacotes que compartilham a mesma filosofia e linguagem de programação. Fazem parte dessa família pacotes como o dplyr, o readr e o ggplot2, entre outros. O dplyr é um pacote com funções para a **manipulação de dados**, O ggplot2 é um pacote poderoso para fazer **gráficos** e o readr permite a importação e exportação de dados.

Uma função básica do Tidyverse é a **função pipe** %>%, que se obtém pressionando as teclas CTRL+SHIFT+M. Essa função pode ser interpretada como “então” e permite escrever códigos na sequência em que ocorrem. Por exemplo:

# para criar e inserir uma variável no banco de dados, ao invés de usar o sistema tradicional, podemos usar a linguagem tidy.  
  
library(tidyverse)  
  
df %>% group\_by(sexo) %>% summarise(média = mean(idade))

## # A tibble: 2 x 2  
## sexo média  
## <fct> <dbl>  
## 1 feminino NA   
## 2 masculino 37.2

df %>% group\_by(sexo) %>% summarise(média = mean(idade, na.rm = TRUE))

## # A tibble: 2 x 2  
## sexo média  
## <fct> <dbl>  
## 1 feminino 32   
## 2 masculino 37.2

str(df)

## 'data.frame': 6 obs. of 15 variables:  
## $ id : chr "s1" "s2" "s3" "s4" ...  
## $ sexo : Factor w/ 2 levels "feminino","masculino": 2 1 2 2 1 2  
## $ idade : num 25 32 78 12 NA 34  
## $ escolaridade: Factor w/ 3 levels "fundamental",..: 3 2 1 1 2 3  
## $ estado\_civil: chr "casado/a" "solteiro/a" "viúvo/a" "solteiro/a" ...  
## $ renda : num 1000 1500 2000 2500 3000 3500  
## $ bonus : num 500 500 500 500 500 500  
## $ renda\_total : num 1500 2000 2500 3000 3500 4000  
## $ v1 :Class 'labelled' num 2 4 3 5 1 2  
## .. .. LABEL: Você costuma se atrasar para as aulas?   
## $ v2 :Class 'labelled' num 2 1 3 1 4 5  
## .. .. LABEL: Quanto você confia na sua capacidade de aprender?   
## $ v3 :Class 'labelled' num 5 3 4 2 4 3  
## .. .. LABEL: Quão bem você se relaciona com outras pessoas na sua escola?   
## $ v4 :Class 'labelled' num 1 5 3 4 2 5  
## .. .. LABEL: Quanto você gosta de sua escola?   
## $ v5 :Class 'labelled' num 3 2 4 1 5 3  
## .. .. LABEL: Quanto você procrastina para estudar?   
## $ renda\_cod : num 1 1 1 2 2 2  
## $ escola\_cod : num 3 2 1 1 2 3

Na sintaxe acima, usamos algumas funções que serão discutidas a seguir, mas é possível ver como escrevemos os códigos usando o pipe. Em português, esse código ficaria assim: pegue o dataframe df, **então** agrupe pela variável sexo, **então** calcule as médias de idade. Esse comando ajuda bastante a escrever códigos mais complexos.

O tidyverse também apresenta diversos recursos para manipulação de dados, entre os quais estão:

| Funções.tidyverse.dplyer. | O.que.faz |
| --- | --- |
| select() | seleciona colunas do dataframe |
| filter() | filtra linhas por categoria de variável |
| slice() | filtra linhas inteiras do dataframe |
| arrange() | reordena as linhas do dataframe |
| mutate() | cria novas colunas no dataframe |
| group\_by() | realiza os comandos que vierem a seguir pelas variáveis selecionadas |

Vamos testar uma por uma, com o nosso dataframe df, começando pela função select(), que seleciona **colunas** do dataframe.

## select()

# SELECT  
  
names(df) # visualisar número das colunas

## [1] "id" "sexo" "idade" "escolaridade" "estado\_civil"  
## [6] "renda" "bonus" "renda\_total" "v1" "v2"   
## [11] "v3" "v4" "v5" "renda\_cod" "escola\_cod"

df %>% names() %>% as.data.frame()

## .  
## 1 id  
## 2 sexo  
## 3 idade  
## 4 escolaridade  
## 5 estado\_civil  
## 6 renda  
## 7 bonus  
## 8 renda\_total  
## 9 v1  
## 10 v2  
## 11 v3  
## 12 v4  
## 13 v5  
## 14 renda\_cod  
## 15 escola\_cod

# pegar df e selecionar as colunas 1,2,4,5 e de 9 a 13  
  
df[ ,c("id","sexo","escolaridade","estado\_civil","v1","v2","v3","v4","v5")]

## id sexo escolaridade estado\_civil v1 v2 v3 v4 v5  
## 1 s1 masculino superior casado/a 2 2 5 1 3  
## 2 s2 feminino médio solteiro/a 4 1 3 5 2  
## 3 s3 masculino fundamental viúvo/a 3 3 4 3 4  
## 4 s4 masculino fundamental solteiro/a 5 1 2 4 1  
## 5 s5 feminino médio casado/a 1 4 4 2 5  
## 6 s6 masculino superior casado/a 2 5 3 5 3

df %>% select(1,2,4,5,9:13) # o comando pipe diz o que é pra fazer com o dataframe

## id sexo escolaridade estado\_civil v1 v2 v3 v4 v5  
## 1 s1 masculino superior casado/a 2 2 5 1 3  
## 2 s2 feminino médio solteiro/a 4 1 3 5 2  
## 3 s3 masculino fundamental viúvo/a 3 3 4 3 4  
## 4 s4 masculino fundamental solteiro/a 5 1 2 4 1  
## 5 s5 feminino médio casado/a 1 4 4 2 5  
## 6 s6 masculino superior casado/a 2 5 3 5 3

df %>% select(id, sexo, v1,v2,v3,v4,v5)

## id sexo v1 v2 v3 v4 v5  
## 1 s1 masculino 2 2 5 1 3  
## 2 s2 feminino 4 1 3 5 2  
## 3 s3 masculino 3 3 4 3 4  
## 4 s4 masculino 5 1 2 4 1  
## 5 s5 feminino 1 4 4 2 5  
## 6 s6 masculino 2 5 3 5 3

# podemos salvar esse novo dataframe com um novo nome. Ao rodar o código abaixo, será criado um objeto df1 no Global Environment  
  
df1 <- df %>% select(1,2,4,5,9:13)  
  
# usar select para deletar uma coluna   
  
df1 %>% select(-4) # deletar a coluna estado\_civil

## id sexo escolaridade v1 v2 v3 v4 v5  
## 1 s1 masculino superior 2 2 5 1 3  
## 2 s2 feminino médio 4 1 3 5 2  
## 3 s3 masculino fundamental 3 3 4 3 4  
## 4 s4 masculino fundamental 5 1 2 4 1  
## 5 s5 feminino médio 1 4 4 2 5  
## 6 s6 masculino superior 2 5 3 5 3

# usar select para reordenar as colunas  
  
df1 %>% select(5:9,1,3,2,4)

## v1 v2 v3 v4 v5 id escolaridade sexo estado\_civil  
## 1 2 2 5 1 3 s1 superior masculino casado/a  
## 2 4 1 3 5 2 s2 médio feminino solteiro/a  
## 3 3 3 4 3 4 s3 fundamental masculino viúvo/a  
## 4 5 1 2 4 1 s4 fundamental masculino solteiro/a  
## 5 1 4 4 2 5 s5 médio feminino casado/a  
## 6 2 5 3 5 3 s6 superior masculino casado/a

## OU  
  
df1 %>% select(5:9,everything())

## v1 v2 v3 v4 v5 id sexo escolaridade estado\_civil  
## 1 2 2 5 1 3 s1 masculino superior casado/a  
## 2 4 1 3 5 2 s2 feminino médio solteiro/a  
## 3 3 3 4 3 4 s3 masculino fundamental viúvo/a  
## 4 5 1 2 4 1 s4 masculino fundamental solteiro/a  
## 5 1 4 4 2 5 s5 feminino médio casado/a  
## 6 2 5 3 5 3 s6 masculino superior casado/a

# Outras possibilidades  
  
df %>% select(contains("sum"))

## data frame with 0 columns and 6 rows

df %>% select(ends\_with("dade"))

## idade escolaridade  
## 1 25 superior  
## 2 32 médio  
## 3 78 fundamental  
## 4 12 fundamental  
## 5 NA médio  
## 6 34 superior

df %>% select(starts\_with("v"))

## v1 v2 v3 v4 v5  
## 1 2 2 5 1 3  
## 2 4 1 3 5 2  
## 3 3 3 4 3 4  
## 4 5 1 2 4 1  
## 5 1 4 4 2 5  
## 6 2 5 3 5 3

# vamos aproveitar e deletar as duas últimas colunas do df  
  
names(df) # ver o nome e o número das colunas, para identificar que as duas últimas colunas são a 14 e a 15.

## [1] "id" "sexo" "idade" "escolaridade" "estado\_civil"  
## [6] "renda" "bonus" "renda\_total" "v1" "v2"   
## [11] "v3" "v4" "v5" "renda\_cod" "escola\_cod"

df %>% select(-14,-15)

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s1 masculino 25 superior casado/a 1000 500 1500 2 2 5  
## 2 s2 feminino 32 médio solteiro/a 1500 500 2000 4 1 3  
## 3 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3 4  
## 4 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5 1 2  
## 5 s5 feminino NA médio casado/a 3000 500 3500 1 4 4  
## 6 s6 masculino 34 superior casado/a 3500 500 4000 2 5 3  
## v4 v5  
## 1 1 3  
## 2 5 2  
## 3 3 4  
## 4 4 1  
## 5 2 5  
## 6 5 3

df %>% select(-c(14,15))

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s1 masculino 25 superior casado/a 1000 500 1500 2 2 5  
## 2 s2 feminino 32 médio solteiro/a 1500 500 2000 4 1 3  
## 3 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3 4  
## 4 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5 1 2  
## 5 s5 feminino NA médio casado/a 3000 500 3500 1 4 4  
## 6 s6 masculino 34 superior casado/a 3500 500 4000 2 5 3  
## v4 v5  
## 1 1 3  
## 2 5 2  
## 3 3 4  
## 4 4 1  
## 5 2 5  
## 6 5 3

df %>% select(-c(renda\_cod,escola\_cod))

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s1 masculino 25 superior casado/a 1000 500 1500 2 2 5  
## 2 s2 feminino 32 médio solteiro/a 1500 500 2000 4 1 3  
## 3 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3 4  
## 4 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5 1 2  
## 5 s5 feminino NA médio casado/a 3000 500 3500 1 4 4  
## 6 s6 masculino 34 superior casado/a 3500 500 4000 2 5 3  
## v4 v5  
## 1 1 3  
## 2 5 2  
## 3 3 4  
## 4 4 1  
## 5 2 5  
## 6 5 3

df <- df %>% select(-c(renda\_cod,escola\_cod))  
  
# se quiser selecionar um grupo de variáveis que começam com as mesmas letras  
  
df %>% select(starts\_with("v"))

## v1 v2 v3 v4 v5  
## 1 2 2 5 1 3  
## 2 4 1 3 5 2  
## 3 3 3 4 3 4  
## 4 5 1 2 4 1  
## 5 1 4 4 2 5  
## 6 2 5 3 5 3

df %>% select(starts\_with(c("v","sum","mean")))

## v1 v2 v3 v4 v5  
## 1 2 2 5 1 3  
## 2 4 1 3 5 2  
## 3 3 3 4 3 4  
## 4 5 1 2 4 1  
## 5 1 4 4 2 5  
## 6 2 5 3 5 3

# ou selecionar variáveis que terminam com as mesmas letras (ou números)  
  
df %>% select(ends\_with("dade"))

## idade escolaridade  
## 1 25 superior  
## 2 32 médio  
## 3 78 fundamental  
## 4 12 fundamental  
## 5 NA médio  
## 6 34 superior

# seleção de variáveis usando operadores booleanos  
  
df %>% select(!ends\_with("dade"))

## id sexo estado\_civil renda bonus renda\_total v1 v2 v3 v4 v5  
## 1 s1 masculino casado/a 1000 500 1500 2 2 5 1 3  
## 2 s2 feminino solteiro/a 1500 500 2000 4 1 3 5 2  
## 3 s3 masculino viúvo/a 2000 500 2500 3 3 4 3 4  
## 4 s4 masculino solteiro/a 2500 500 3000 5 1 2 4 1  
## 5 s5 feminino casado/a 3000 500 3500 1 4 4 2 5  
## 6 s6 masculino casado/a 3500 500 4000 2 5 3 5 3

df %>% select(where(is.factor))

## sexo escolaridade  
## 1 masculino superior  
## 2 feminino médio  
## 3 masculino fundamental  
## 4 masculino fundamental  
## 5 feminino médio  
## 6 masculino superior

df %>% select(where(is.numeric) & starts\_with(c("s")))

## data frame with 0 columns and 6 rows

df %>% select(where(is.numeric) & starts\_with(c("s","v")))

## v1 v2 v3 v4 v5  
## 1 2 2 5 1 3  
## 2 4 1 3 5 2  
## 3 3 3 4 3 4  
## 4 5 1 2 4 1  
## 5 1 4 4 2 5  
## 6 2 5 3 5 3

## filter()

Se a função select() permite a seleção de colunas de um dataframe, a função filter() permite a seleção de **linhas**.

## se quisermos filtrar somente os participantes do sexo masculino  
  
df %>% filter(sexo == "masculino")

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s1 masculino 25 superior casado/a 1000 500 1500 2 2 5  
## 2 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3 4  
## 3 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5 1 2  
## 4 s6 masculino 34 superior casado/a 3500 500 4000 2 5 3  
## v4 v5  
## 1 1 3  
## 2 3 4  
## 3 4 1  
## 4 5 3

## pode-se acrescentar outras variáveis para filtrar  
## por exemplo, filtrar participantes do sexo masculino, com nivel fundamental de escolaridade  
  
df %>% filter(sexo == "masculino",escolaridade == "fundamental")

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3 4  
## 2 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5 1 2  
## v4 v5  
## 1 3 4  
## 2 4 1

## ou ainda participantes do sexo masculino, do esnisno fundamental e solteiro/a  
df %>% filter(sexo == "masculino",escolaridade == "fundamental",estado\_civil == "solteiro/a")

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5 1 2  
## v4 v5  
## 1 4 1

## se quisermos retirar do banco todos os participantes com ensino fundamental.  
## observe que o ponto de exclamação no código é uma espécie de "tudo, menos..."  
df %>% filter(!escolaridade == "fundamental")

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s1 masculino 25 superior casado/a 1000 500 1500 2 2 5  
## 2 s2 feminino 32 médio solteiro/a 1500 500 2000 4 1 3  
## 3 s5 feminino NA médio casado/a 3000 500 3500 1 4 4  
## 4 s6 masculino 34 superior casado/a 3500 500 4000 2 5 3  
## v4 v5  
## 1 1 3  
## 2 5 2  
## 3 2 5  
## 4 5 3

df %>% filter(!escolaridade == "fundamental",renda>=3000)

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s5 feminino NA médio casado/a 3000 500 3500 1 4 4  
## 2 s6 masculino 34 superior casado/a 3500 500 4000 2 5 3  
## v4 v5  
## 1 2 5  
## 2 5 3

## ou, se quiser selecionar participantes com renda\_total superior ou igual a R$ 2000,00.  
df %>% filter(renda\_total >= 2000)

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s2 feminino 32 médio solteiro/a 1500 500 2000 4 1 3  
## 2 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3 4  
## 3 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5 1 2  
## 4 s5 feminino NA médio casado/a 3000 500 3500 1 4 4  
## 5 s6 masculino 34 superior casado/a 3500 500 4000 2 5 3  
## v4 v5  
## 1 5 2  
## 2 3 4  
## 3 4 1  
## 4 2 5  
## 5 5 3

## OBS.: lembrando que se quisermos salvar a filtragem em um novo dataframe...  
df\_renda\_maior2000 <- df %>% filter(renda\_total >= 2000)

As funções select() e filter() são muito úteis para limpeza do banco de dados. Por exemplo, quando baixamos um banco de dados do google formulário, ele vem com as informações de data e hora que o participante respondeu o questionário, com as respoostas ao TCLE, que normalmente não iremos utilizar nas análises. Então, podemos deletá-las usando a função select(). Outra aplicação muito comum é para criar um *subset* do arquivo base, que contenha somente os itens de um dos instrumentos, para análise de suas propriedades psicométrias.  
De forma semelhante, podemos usar a função filter() para eliminar sujeitos que não atendem aos critérios de inclusão da pesquisa. Ou selecionar um *subset* do banco de dados, que contenha os sujeitos de interesse.

## slice()

Uma opção para selecionar casos com base na linha inteira é a função slice().

# selecionar linhas inteiras  
df %>% slice(1:3)

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s1 masculino 25 superior casado/a 1000 500 1500 2 2 5  
## 2 s2 feminino 32 médio solteiro/a 1500 500 2000 4 1 3  
## 3 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3 4  
## v4 v5  
## 1 1 3  
## 2 5 2  
## 3 3 4

df %>% slice(1,4,6)

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s1 masculino 25 superior casado/a 1000 500 1500 2 2 5  
## 2 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5 1 2  
## 3 s6 masculino 34 superior casado/a 3500 500 4000 2 5 3  
## v4 v5  
## 1 1 3  
## 2 4 1  
## 3 5 3

# selecionar por exclusão  
df %>% slice(-1)

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s2 feminino 32 médio solteiro/a 1500 500 2000 4 1 3  
## 2 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3 4  
## 3 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5 1 2  
## 4 s5 feminino NA médio casado/a 3000 500 3500 1 4 4  
## 5 s6 masculino 34 superior casado/a 3500 500 4000 2 5 3  
## v4 v5  
## 1 5 2  
## 2 3 4  
## 3 4 1  
## 4 2 5  
## 5 5 3

df %>% slice(-c(1,6))

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s2 feminino 32 médio solteiro/a 1500 500 2000 4 1 3  
## 2 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3 4  
## 3 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5 1 2  
## 4 s5 feminino NA médio casado/a 3000 500 3500 1 4 4  
## v4 v5  
## 1 5 2  
## 2 3 4  
## 3 4 1  
## 4 2 5

df %>% slice(-1,-6)

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s2 feminino 32 médio solteiro/a 1500 500 2000 4 1 3  
## 2 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3 4  
## 3 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5 1 2  
## 4 s5 feminino NA médio casado/a 3000 500 3500 1 4 4  
## v4 v5  
## 1 5 2  
## 2 3 4  
## 3 4 1  
## 4 2 5

# sortear uma proporção do banco de dados  
df %>% slice\_sample(prop = .35)

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s6 masculino 34 superior casado/a 3500 500 4000 2 5 3  
## 2 s5 feminino NA médio casado/a 3000 500 3500 1 4 4  
## v4 v5  
## 1 5 3  
## 2 2 5

df %>% slice\_sample(n=3)

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s5 feminino NA médio casado/a 3000 500 3500 1 4 4  
## 2 s6 masculino 34 superior casado/a 3500 500 4000 2 5 3  
## 3 s1 masculino 25 superior casado/a 1000 500 1500 2 2 5  
## v4 v5  
## 1 2 5  
## 2 5 3  
## 3 1 3

## arrange()

Por sua vez, a função arrange(), ordena os dados por um critério.

# Ordenar df por idade  
df %>% arrange(idade)

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5 1 2  
## 2 s1 masculino 25 superior casado/a 1000 500 1500 2 2 5  
## 3 s2 feminino 32 médio solteiro/a 1500 500 2000 4 1 3  
## 4 s6 masculino 34 superior casado/a 3500 500 4000 2 5 3  
## 5 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3 4  
## 6 s5 feminino NA médio casado/a 3000 500 3500 1 4 4  
## v4 v5  
## 1 4 1  
## 2 1 3  
## 3 5 2  
## 4 5 3  
## 5 3 4  
## 6 2 5

# Ordenar df por ordem decrescente de renda\_total  
df %>% arrange(desc(renda\_total))

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s6 masculino 34 superior casado/a 3500 500 4000 2 5 3  
## 2 s5 feminino NA médio casado/a 3000 500 3500 1 4 4  
## 3 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5 1 2  
## 4 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3 4  
## 5 s2 feminino 32 médio solteiro/a 1500 500 2000 4 1 3  
## 6 s1 masculino 25 superior casado/a 1000 500 1500 2 2 5  
## v4 v5  
## 1 5 3  
## 2 2 5  
## 3 4 1  
## 4 3 4  
## 5 5 2  
## 6 1 3

# É possível inserir mais de uma variável, por exemplo, sexo e renda\_total  
df %>% arrange(sexo,desc(renda\_total))

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s5 feminino NA médio casado/a 3000 500 3500 1 4 4  
## 2 s2 feminino 32 médio solteiro/a 1500 500 2000 4 1 3  
## 3 s6 masculino 34 superior casado/a 3500 500 4000 2 5 3  
## 4 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5 1 2  
## 5 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3 4  
## 6 s1 masculino 25 superior casado/a 1000 500 1500 2 2 5  
## v4 v5  
## 1 2 5  
## 2 5 2  
## 3 5 3  
## 4 4 1  
## 5 3 4  
## 6 1 3

A função mutate() permite criar variáveis dentro de um dataframe. É muito útil para calcular pontuações a partir de itens, por exemplo, ou para fazer transformações de outras variáveis.  
## mutate

# calcular as pontuações das pessoas nos itens de v1 a v5.  
df %>% mutate(escore = v1+v2+v3+v4+v5)

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s1 masculino 25 superior casado/a 1000 500 1500 2 2 5  
## 2 s2 feminino 32 médio solteiro/a 1500 500 2000 4 1 3  
## 3 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3 4  
## 4 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5 1 2  
## 5 s5 feminino NA médio casado/a 3000 500 3500 1 4 4  
## 6 s6 masculino 34 superior casado/a 3500 500 4000 2 5 3  
## v4 v5 escore  
## 1 1 3 13  
## 2 5 2 15  
## 3 3 4 17  
## 4 4 1 13  
## 5 2 5 16  
## 6 5 3 18

df %>% mutate(escore = (v1+v2+v3+v4+v5)/5)

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s1 masculino 25 superior casado/a 1000 500 1500 2 2 5  
## 2 s2 feminino 32 médio solteiro/a 1500 500 2000 4 1 3  
## 3 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3 4  
## 4 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5 1 2  
## 5 s5 feminino NA médio casado/a 3000 500 3500 1 4 4  
## 6 s6 masculino 34 superior casado/a 3500 500 4000 2 5 3  
## v4 v5 escore  
## 1 1 3 2.6  
## 2 5 2 3.0  
## 3 3 4 3.4  
## 4 4 1 2.6  
## 5 2 5 3.2  
## 6 5 3 3.6

# se quisermos salvar a nova variável, então o código seria...  
df <- df %>% mutate(escore = v1+v2+v3+v4+v5)  
  
# remover label da variável escore  
escore <- unvr(df$escore)  
df$escore <- escore  
glimpse(df)

## Rows: 6  
## Columns: 14  
## $ id <chr> "s1", "s2", "s3", "s4", "s5", "s6"  
## $ sexo <fct> masculino, feminino, masculino, masculino, feminino, masc~  
## $ idade <dbl> 25, 32, 78, 12, NA, 34  
## $ escolaridade <fct> superior, médio, fundamental, fundamental, médio, superior  
## $ estado\_civil <chr> "casado/a", "solteiro/a", "viúvo/a", "solteiro/a", "casad~  
## $ renda <dbl> 1000, 1500, 2000, 2500, 3000, 3500  
## $ bonus <dbl> 500, 500, 500, 500, 500, 500  
## $ renda\_total <dbl> 1500, 2000, 2500, 3000, 3500, 4000  
## $ v1 <labelled> 2, 4, 3, 5, 1, 2  
## $ v2 <labelled> 2, 1, 3, 1, 4, 5  
## $ v3 <labelled> 5, 3, 4, 2, 4, 3  
## $ v4 <labelled> 1, 5, 3, 4, 2, 5  
## $ v5 <labelled> 3, 2, 4, 1, 5, 3  
## $ escore <dbl> 13, 15, 17, 13, 16, 18

## group\_by

A função group\_by agrupa os resultados de uma operação e, geralmente, é usada juntamente com alguma outra operação, como por exemplo, a função summarise().

# calcular as médias da variável escore, por sexo (grupo\_by(sexo))  
df %>% group\_by(sexo) %>% count(escolaridade)

## # A tibble: 3 x 3  
## # Groups: sexo [2]  
## sexo escolaridade n  
## <fct> <fct> <int>  
## 1 feminino médio 2  
## 2 masculino fundamental 2  
## 3 masculino superior 2

df %>% group\_by(sexo) %>% summarise(mean(escore))

## # A tibble: 2 x 2  
## sexo `mean(escore)`  
## <fct> <dbl>  
## 1 feminino 15.5  
## 2 masculino 15.2

df %>% group\_by(sexo,escolaridade) %>% summarise(mean(escore))

## # A tibble: 3 x 3  
## # Groups: sexo [2]  
## sexo escolaridade `mean(escore)`  
## <fct> <fct> <dbl>  
## 1 feminino médio 15.5  
## 2 masculino fundamental 15   
## 3 masculino superior 15.5

## rename

Às vezes, é necessário renomear uma ou mais variáveis de um dataframe. Para isso, podemos usar a função rename() do dyplr.

df %>% rename(item1 = v1,  
 item2 = v2,  
 item3 = v3,  
 item4 = v4,  
 item5 = v5)

## id sexo idade escolaridade estado\_civil renda bonus renda\_total item1  
## 1 s1 masculino 25 superior casado/a 1000 500 1500 2  
## 2 s2 feminino 32 médio solteiro/a 1500 500 2000 4  
## 3 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3  
## 4 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5  
## 5 s5 feminino NA médio casado/a 3000 500 3500 1  
## 6 s6 masculino 34 superior casado/a 3500 500 4000 2  
## item2 item3 item4 item5 escore  
## 1 2 5 1 3 13  
## 2 1 3 5 2 15  
## 3 3 4 3 4 17  
## 4 1 2 4 1 13  
## 5 4 4 2 5 16  
## 6 5 3 5 3 18

## Combinação de variáveis (left\_join, right\_join, inner\_join, full\_join, semi\_join, anti\_join)

Num primeiro caso, vamos combinar dois data frames com o mesmo número de linhas e colunas. Para isso, é necessário ter pelo menos uma variável em comum nos dois bancos de dados. Esssa variável será usada pela função para combinar os dois dataframes.

## criando df2 para combinar com df  
df2 <-   
data.frame(id = c("s1","s2","s3","s4","s5","s6"),  
 Trabalho = c(1,1,0,0,1,1),  
 Filhos = c(0,1,1,0,1,1),  
 renda\_total = c(1500,2000,2500,3000,3500,4000))  
  
left\_join(df,df2) # Para salvar o novo dataframe, teríamos que apontar a função para um novo objeto.

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s1 masculino 25 superior casado/a 1000 500 1500 2 2 5  
## 2 s2 feminino 32 médio solteiro/a 1500 500 2000 4 1 3  
## 3 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3 4  
## 4 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5 1 2  
## 5 s5 feminino NA médio casado/a 3000 500 3500 1 4 4  
## 6 s6 masculino 34 superior casado/a 3500 500 4000 2 5 3  
## v4 v5 escore Trabalho Filhos  
## 1 1 3 13 1 0  
## 2 5 2 15 1 1  
## 3 3 4 17 0 1  
## 4 4 1 13 0 0  
## 5 2 5 16 1 1  
## 6 5 3 18 1 1

Note que a função reconheceu duas variáveis em comum (id e renda\_total) e sinalizou isso dizendo Joining, by = c("id", "renda\_total") no início dos resultados. Mas, muitas vezes a situação é mais complexa. Podemos ter um segundo dataframe com informações adicionais de apenas **ALGUNS** participantes que estão no primeiro dataframe. Além disso, podemos ter participantes adicionais, que não estão no primeiro dataframe.

# Não temos mais os participantes s2 e s4 e temos outros participantes (s7 a s10), que não estão em df  
df2 <-   
data.frame(id = c("s1","s3","s5","s6","s7","s8","s9","s10"),  
 Trabalho = c(1,0,1,1,1,0,1,0),  
 Filhos = c(0,1,1,1,0,0,1,1))  
  
left\_join(df,df2)

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s1 masculino 25 superior casado/a 1000 500 1500 2 2 5  
## 2 s2 feminino 32 médio solteiro/a 1500 500 2000 4 1 3  
## 3 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3 4  
## 4 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5 1 2  
## 5 s5 feminino NA médio casado/a 3000 500 3500 1 4 4  
## 6 s6 masculino 34 superior casado/a 3500 500 4000 2 5 3  
## v4 v5 escore Trabalho Filhos  
## 1 1 3 13 1 0  
## 2 5 2 15 NA NA  
## 3 3 4 17 0 1  
## 4 4 1 13 NA NA  
## 5 2 5 16 1 1  
## 6 5 3 18 1 1

Observe que somente os **DADOS EM COMUM** foram para o novo dataframe. Ou seja, participantes que não constavam em df não foram acrescentados e informações não disponíveis em df2 sobre participantes de df foram registradas como NA. A função chama-se left\_join exatamente por tomar o dataframe à esquerda (o primeiro) como referência para fazer as alterações.  
Se quisermos usar o segundo dataframe como referência (df2), podemos usar a função right\_join

right\_join(df,df2)

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2  
## 1 s1 masculino 25 superior casado/a 1000 500 1500 2 2  
## 2 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3  
## 3 s5 feminino NA médio casado/a 3000 500 3500 1 4  
## 4 s6 masculino 34 superior casado/a 3500 500 4000 2 5  
## 5 s7 <NA> NA <NA> <NA> NA NA NA NA NA  
## 6 s8 <NA> NA <NA> <NA> NA NA NA NA NA  
## 7 s9 <NA> NA <NA> <NA> NA NA NA NA NA  
## 8 s10 <NA> NA <NA> <NA> NA NA NA NA NA  
## v3 v4 v5 escore Trabalho Filhos  
## 1 5 1 3 13 1 0  
## 2 4 3 4 17 0 1  
## 3 4 2 5 16 1 1  
## 4 3 5 3 18 1 1  
## 5 NA NA NA NA 1 0  
## 6 NA NA NA NA 0 0  
## 7 NA NA NA NA 1 1  
## 8 NA NA NA NA 0 1

Caso se queira apenas as informações que constem **em ambos os dataframes**, pode-se usar a função inner\_join.

inner\_join(df,df2)

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s1 masculino 25 superior casado/a 1000 500 1500 2 2 5  
## 2 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3 4  
## 3 s5 feminino NA médio casado/a 3000 500 3500 1 4 4  
## 4 s6 masculino 34 superior casado/a 3500 500 4000 2 5 3  
## v4 v5 escore Trabalho Filhos  
## 1 1 3 13 1 0  
## 2 3 4 17 0 1  
## 3 2 5 16 1 1  
## 4 5 3 18 1 1

Caso se queira a **JUNÇÃO COMPLETA DOS DOIS DATAFRAMES**, pode-se usar a função full\_join.

full\_join(df,df2)

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2  
## 1 s1 masculino 25 superior casado/a 1000 500 1500 2 2  
## 2 s2 feminino 32 médio solteiro/a 1500 500 2000 4 1  
## 3 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3  
## 4 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5 1  
## 5 s5 feminino NA médio casado/a 3000 500 3500 1 4  
## 6 s6 masculino 34 superior casado/a 3500 500 4000 2 5  
## 7 s7 <NA> NA <NA> <NA> NA NA NA NA NA  
## 8 s8 <NA> NA <NA> <NA> NA NA NA NA NA  
## 9 s9 <NA> NA <NA> <NA> NA NA NA NA NA  
## 10 s10 <NA> NA <NA> <NA> NA NA NA NA NA  
## v3 v4 v5 escore Trabalho Filhos  
## 1 5 1 3 13 1 0  
## 2 3 5 2 15 NA NA  
## 3 4 3 4 17 0 1  
## 4 2 4 1 13 NA NA  
## 5 4 2 5 16 1 1  
## 6 3 5 3 18 1 1  
## 7 NA NA NA NA 1 0  
## 8 NA NA NA NA 0 0  
## 9 NA NA NA NA 1 1  
## 10 NA NA NA NA 0 1

Caso a variável comum esteja com nomes diferentes nos dois dataframes, basta acrescentar essa informação ao comando.

df2 <-   
data.frame(Participantes = c("s1","s3","s5","s6","s7","s8","s9","s10"),  
 Trabalho = c(1,0,1,1,1,0,1,0),  
 Filhos = c(0,1,1,1,0,0,1,1))  
  
left\_join(df,df2, by = c("id" = "Participantes"))

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s1 masculino 25 superior casado/a 1000 500 1500 2 2 5  
## 2 s2 feminino 32 médio solteiro/a 1500 500 2000 4 1 3  
## 3 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3 4  
## 4 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5 1 2  
## 5 s5 feminino NA médio casado/a 3000 500 3500 1 4 4  
## 6 s6 masculino 34 superior casado/a 3500 500 4000 2 5 3  
## v4 v5 escore Trabalho Filhos  
## 1 1 3 13 1 0  
## 2 5 2 15 NA NA  
## 3 3 4 17 0 1  
## 4 4 1 13 NA NA  
## 5 2 5 16 1 1  
## 6 5 3 18 1 1

A função semi\_join pode ser empregada para filtrar as observações (linhas) de df que possuem correspondência em df2.

semi\_join(df,df2, by = c("id" = "Participantes"))

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s1 masculino 25 superior casado/a 1000 500 1500 2 2 5  
## 2 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3 4  
## 3 s5 feminino NA médio casado/a 3000 500 3500 1 4 4  
## 4 s6 masculino 34 superior casado/a 3500 500 4000 2 5 3  
## v4 v5 escore  
## 1 1 3 13  
## 2 3 4 17  
## 3 2 5 16  
## 4 5 3 18

Note que nenhuma variável é acrescentada, mas o dataframe é reduzido para apenas as observações de df que apresentam todas as informações em df2. O contrário também pode ser obtido, ou seja, as informações de df que NÃO APRESENTAM CORRESPONDÊNCIA em df2. Para isso, devemos usar a função anti\_join.

anti\_join(df,df2, by = c("id" = "Participantes"))

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s2 feminino 32 médio solteiro/a 1500 500 2000 4 1 3  
## 2 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5 1 2  
## v4 v5 escore  
## 1 5 2 15  
## 2 4 1 13

## Combinação de observações (union, setdiff, intersect, bind\_rows)

O dataframe df3 contém cinco observações (s6 a s10) compatíveis com as variáveis de df. sendo que as informações do sujeito s6 estão repetidas em ambos os dataframes.

Para unir todas as observações de df e df3 podemos usar a função union.

union(df,df3)

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2  
## 1 s1 masculino 25 superior casado/a 1000 500 1500 2 2  
## 2 s2 feminino 32 médio solteiro/a 1500 500 2000 4 1  
## 3 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3  
## 4 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5 1  
## 5 s5 feminino NA médio casado/a 3000 500 3500 1 4  
## 6 s6 masculino 34 superior casado/a 3500 500 4000 2 5  
## 7 s7 feminino 55 superior casado/a 2500 500 3000 2 3  
## 8 s8 masculino 43 médio casado/a 1000 500 1500 1 2  
## 9 s9 masculino 39 fundamental casado/a 1500 500 2000 3 1  
## 10 s10 feminino 15 médio casado/a 4000 500 4500 1 1  
## v3 v4 v5 escore  
## 1 5 1 3 13  
## 2 3 5 2 15  
## 3 4 3 4 17  
## 4 2 4 1 13  
## 5 4 2 5 16  
## 6 3 5 3 18  
## 7 2 2 1 10  
## 8 2 2 2 9  
## 9 1 3 4 12  
## 10 5 1 5 13

Para selecionar somente as observações em comum nos dois dataframes, podemos usar a função intersect.

intersect(df,df3)

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s6 masculino 34 superior casado/a 3500 500 4000 2 5 3  
## v4 v5 escore  
## 1 5 3 18

Para selecionar somente as observações únicas de df, podemos usar a função setdiff.

setdiff(df,df3)

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s1 masculino 25 superior casado/a 1000 500 1500 2 2 5  
## 2 s2 feminino 32 médio solteiro/a 1500 500 2000 4 1 3  
## 3 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3 4  
## 4 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5 1 2  
## 5 s5 feminino NA médio casado/a 3000 500 3500 1 4 4  
## v4 v5 escore  
## 1 1 3 13  
## 2 5 2 15  
## 3 3 4 17  
## 4 4 1 13  
## 5 2 5 16

Para unir os dataframes df e df3, mas colocando uma identificação de onde veio cada observação, podemos usar bind\_rows.

bind\_rows(df,df3,.id="grupo")

## grupo id sexo idade escolaridade estado\_civil renda bonus renda\_total  
## 1 1 s1 masculino 25 superior casado/a 1000 500 1500  
## 2 1 s2 feminino 32 médio solteiro/a 1500 500 2000  
## 3 1 s3 masculino 78 fundamental viúvo/a 2000 500 2500  
## 4 1 s4 masculino 12 fundamental solteiro/a 2500 500 3000  
## 5 1 s5 feminino NA médio casado/a 3000 500 3500  
## 6 1 s6 masculino 34 superior casado/a 3500 500 4000  
## 7 2 s6 masculino 34 superior casado/a 3500 500 4000  
## 8 2 s7 feminino 55 superior casado/a 2500 500 3000  
## 9 2 s8 masculino 43 médio casado/a 1000 500 1500  
## 10 2 s9 masculino 39 fundamental casado/a 1500 500 2000  
## 11 2 s10 feminino 15 médio casado/a 4000 500 4500  
## v1 v2 v3 v4 v5 escore  
## 1 2 2 5 1 3 13  
## 2 4 1 3 5 2 15  
## 3 3 3 4 3 4 17  
## 4 5 1 2 4 1 13  
## 5 1 4 4 2 5 16  
## 6 2 5 3 5 3 18  
## 7 2 5 3 5 3 18  
## 8 2 3 2 2 1 10  
## 9 1 2 2 2 2 9  
## 10 3 1 1 3 4 12  
## 11 1 1 5 1 5 13

Note, porém, que a saída incluiu duas vezes o sujeito s06, porque a função bind\_rows simplesmente une os dois dataframes, sem fazer nenhum tipo de concatenação entre eles. ## pivot\_longer e pivot\_wider (funções tidyr)

Às vezes precisamos fazer transformações no formato do banco de dados para fazer alguma análise estatística ou gráfico. Precisamos transformar as linhas em colunas ou as colunas em linhas. Esses dois comandos do dplyr ajudam a realizar esses procedimentos.

# ACRESCENTAR INFORMAÇÃO

## Exercícios

dim(df) # ver número de linhas e colunas do dataframe df

## [1] 6 14

imdb <- readr::read\_rds("imdb.rds")  
str(imdb)

## tibble [3,807 x 15] (S3: tbl\_df/tbl/data.frame)  
## $ titulo : chr [1:3807] "Avatar " "Pirates of the Caribbean: At World's End " "The Dark Knight Rises " "John Carter " ...  
## $ ano : int [1:3807] 2009 2007 2012 2012 2007 2010 2015 2016 2006 2006 ...  
## $ diretor : chr [1:3807] "James Cameron" "Gore Verbinski" "Christopher Nolan" "Andrew Stanton" ...  
## $ duracao : int [1:3807] 178 169 164 132 156 100 141 183 169 151 ...  
## $ cor : chr [1:3807] "Color" "Color" "Color" "Color" ...  
## $ generos : chr [1:3807] "Action|Adventure|Fantasy|Sci-Fi" "Action|Adventure|Fantasy" "Action|Thriller" "Action|Adventure|Sci-Fi" ...  
## $ pais : chr [1:3807] "USA" "USA" "USA" "USA" ...  
## $ classificacao : chr [1:3807] "A partir de 13 anos" "A partir de 13 anos" "A partir de 13 anos" "A partir de 13 anos" ...  
## $ orcamento : int [1:3807] 237000000 300000000 250000000 263700000 258000000 260000000 250000000 250000000 209000000 225000000 ...  
## $ receita : int [1:3807] 760505847 309404152 448130642 73058679 336530303 200807262 458991599 330249062 200069408 423032628 ...  
## $ nota\_imdb : num [1:3807] 7.9 7.1 8.5 6.6 6.2 7.8 7.5 6.9 6.1 7.3 ...  
## $ likes\_facebook: int [1:3807] 33000 0 164000 24000 0 29000 118000 197000 0 5000 ...  
## $ ator\_1 : chr [1:3807] "CCH Pounder" "Johnny Depp" "Tom Hardy" "Daryl Sabara" ...  
## $ ator\_2 : chr [1:3807] "Joel David Moore" "Orlando Bloom" "Christian Bale" "Samantha Morton" ...  
## $ ator\_3 : chr [1:3807] "Wes Studi" "Jack Davenport" "Joseph Gordon-Levitt" "Polly Walker" ...

# crie uma variável id para todos os sujeitos do dataframe  
  
# selecione as variáveis título, duração, cor e gênero dos 1000 primeiros casos do banco de dados  
  
# monte um dataframe somente com os filmes do diretor Clint Eastwood  
  
# reorganize o banco de dados com filmes do diretor Clint Eastwood por ordem decrescente de receita  
  
# selecione uma amostra aleatória do dataframe imdb com 45% dos filmes relacionados.  
  
# crie uma tabela com as médias de orçamento e de receita, por diretor.

# Importar e exportar arquivos de/para outros formatos

Há vários pacotes para importação de arquivos elaborados em outros programas, como o Excel e o SPSS, por exemplo, que são muito comuns em pesquisas em Psicologia. De forma semelhante, às vezes, analisamos os dados no R/RStudio, salvamos a planilha em um dataframe, mas gostaríamos de exportar essa planilha para outros formatos, para serem lidos por outros programas, por exemplo. A seguir, veremos como fazer isso…

# Exportar arquivos  
  
# Carregar os pacotes readr  
if(!require(readr)) install.packages("readr")  
library(readr)  
  
## Usando função da base  
  
saveRDS(df,"df.rds") # formato reconhecido pelo R

## Usando funções do pacote utils  
  
# write.csv(df1, # dataframe a ser exportado  
# "df1.csv", # nome a ser dado ao dataframe exportado  
# sep = ",", # caractere que separa as colunas  
# na = "NA", # como codificar dados faltantes  
# dec = ".", # que caractere usar para separar casas decimais  
# col.names = TRUE) # informar se Há (TRUE) ou não (FALSE) nome das colunas no dataframe  
  
# write.csv2(df1, # semelhante ao comando anterior, mas assumindo sep = ";" e dec = ","  
# "df1.csv",   
# sep = ";",   
# na = "NA",   
# dec = ",",   
# col.names = TRUE)  
  
### OBS.: Essas funções acima podem ser resumidas para...  
write.csv(df1, "df1.csv")  
write.csv2(df1, "df1.csv")

## Usando funções do pacote readr  
  
write\_csv(  
 df1,  
 "df1.csv",  
 na = "NA",  
 col\_names = TRUE)  
  
write\_csv2(  
 df1,  
 "df1.csv",  
 na = "NA",  
 col\_names = TRUE)  
  
write\_excel\_csv(  
 df1,  
 "df1.csv",  
 na = "NA",  
 col\_names = TRUE,  
 delim = ",")  
  
write\_excel\_csv2(  
 df1,  
 "df1.csv",  
 na = "NA",  
 col\_names = TRUE,  
 delim = ";")

# Importando arquivos em .csv  
  
## Usando o pacote utils  
  
#### read.csv(df.csv, header = TRUE, sep = ",", dec = ".", encoding = "UTF-8")  
read.csv2("df.csv", # nome do arquivo a ser importado (OBS. deve estar no diretório de trabalho)  
 header = TRUE, # TRUE, para arquivos com nomes das colunas  
 sep = ";", # caractere separador das colunas  
 dec = ",", # caractere separador de casas decimais.  
 encoding = "UTF-8") # usar codificação UTF-8 para que os acentos, etc. venham de forma correta.

## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1 v2 v3  
## 1 s1 masculino 25 superior casado/a 1000 500 1500 2 4 1  
## 2 s2 feminino 32 médio solteiro/a 1500 500 2000 4 5 3  
## 3 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3 3 2  
## 4 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5 5 4  
## 5 s5 feminino NA médio casado/a 3000 500 3500 1 2 2  
## 6 s6 masculino 34 superior casado/a 3500 500 4000 2 1 3  
## v4 v5  
## 1 5 3  
## 2 1 2  
## 3 3 4  
## 4 2 1  
## 5 4 5  
## 6 1 3

## Usando o pacote readr  
  
read\_csv2("df.csv", col\_names = TRUE, na = "NA",skip\_empty\_rows = TRUE)

## # A tibble: 6 x 13  
## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1  
## <chr> <chr> <dbl> <chr> <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 s1 masculino 25 superior casado/a 1000 500 1500 2  
## 2 s2 feminino 32 médio solteiro/a 1500 500 2000 4  
## 3 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3  
## 4 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5  
## 5 s5 feminino NA médio casado/a 3000 500 3500 1  
## 6 s6 masculino 34 superior casado/a 3500 500 4000 2  
## # ... with 4 more variables: v2 <dbl>, v3 <dbl>, v4 <dbl>, v5 <dbl>

## Usando o pacote readxl para importar arquivos em excel  
library(readxl)  
  
read\_excel("df.xlsx", col\_names = TRUE)

## # A tibble: 6 x 13  
## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1  
## <chr> <chr> <dbl> <chr> <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 s1 masculino 25 superior casado/a 1000 500 1500 2  
## 2 s2 feminino 32 médio solteiro/a 1500 500 2000 4  
## 3 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3  
## 4 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5  
## 5 s5 feminino NA médio casado/a 3000 500 3500 1  
## 6 s6 masculino 34 superior casado/a 3500 500 4000 2  
## # ... with 4 more variables: v2 <dbl>, v3 <dbl>, v4 <dbl>, v5 <dbl>

## Usando o pacote haven para importar arquivos do SPSS  
  
#install.packages("haven")  
library(haven)  
  
read\_sav("df.sav")

## # A tibble: 6 x 13  
## id sexo idade escolaridade estado\_civil renda bonus renda\_total v1  
## <chr> <chr> <dbl> <chr> <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 s1 masculino 25 superior casado/a 1000 500 1500 2  
## 2 s2 feminino 32 médio solteiro/a 1500 500 2000 4  
## 3 s3 masculino 78 fundamental viúvo/a 2000 500 2500 3  
## 4 s4 masculino 12 fundamental solteiro/a 2500 500 3000 5  
## 5 s5 feminino NA médio casado/a 3000 500 3500 1  
## 6 s6 masculino 34 superior casado/a 3500 500 4000 2  
## # ... with 4 more variables: v2 <dbl>, v3 <dbl>, v4 <dbl>, v5 <dbl>

# Funções estatísticas básicas

É possível falzer algumas operações matemáticas e de estatística descritiva, com as funções básicas do R. Algumas vezes temos que calcular pontuações a partir da somatória ou da média dos valores atribuídos pelos respondentes aos itens.

## rowSums() e rowMeans()

## calcular pontuações das pessoas em um conjunto de itens.  
## vamos supor que v1, v2 e v5 formam a pontuação em um fator  
## e que v3 e v4 formam a pontuação em outro fator  
  
## se quisermos calcular a somatória das pontuações nesses dois fatores, podemos usar:  
df %>% select(v1,v2,v5) %>% rowSums()

## [1] 7 7 10 7 10 10

## para salvar esses resultados em uma nova coluna no dataframe df, basta usar df$nome  
df$sum1 <- df %>% select(v1,v2,v5) %>% rowSums()  
df$sum2 <- df %>% select(v3,v4) %>% rowSums()  
  
## podemos calcular essas pontuações pela média nos itens ao invés da somatória  
df$mean1 <- df %>% select(v1,v2,v5) %>% rowMeans() %>% round(digits = 1)  
df$mean2 <- df %>% select(v3,v4) %>% rowMeans()

## count()

# contar casos dentro de uma variável.   
# Função útil, por exemplo, para descrever a amostra.  
# dependendo de como se escreve essa função ela dá um tipo de informação.  
  
count(df) # se colocar apenas o nome do dataframe, retorna o número de casos total

## n  
## 1 6

count(df,sexo) # especificando uma varíável, retorna o número de casos dentro da variável

## sexo n  
## 1 feminino 2  
## 2 masculino 4

count(df,sexo,escolaridade) # comando igual a df %>% group\_by(sexo) %>% count(escolaridade)

## sexo escolaridade n  
## 1 feminino médio 2  
## 2 masculino fundamental 2  
## 3 masculino superior 2

# para acrescentar uma coluna com as respectivas porcentagens  
df %>% count(sexo) %>% mutate(porc = n/sum(n)\*100)

## sexo n porc  
## 1 feminino 2 33.33333  
## 2 masculino 4 66.66667

df %>% count(sexo,escolaridade) %>% mutate(porc = n/sum(n)\*100)

## sexo escolaridade n porc  
## 1 feminino médio 2 33.33333  
## 2 masculino fundamental 2 33.33333  
## 3 masculino superior 2 33.33333

## Estatísticas de tendência central e de disperção

A função rowMeans() calcula a média de algumas variáveis em uma **linha**.  
Se quisermos calcular a média e o desvio padrão dos valores de uma **coluna** (variável), temos que usar as funções mean() e sd().  
Outras medidas de tendência central são a mediana e a moda .

# média das pontuações as variáveis "idade", "renda\_total", "escore".  
mean(df$idade)

## [1] NA

# ao rodar essa análise, notei que a variável "idade" está como character  
# ou seja, os valores numéricos estão sendo entendidos como texto  
# então, precisamos fazer uma transformação dessa variável em numérica.  
  
df$idade <- as.numeric(as.character(df$idade))  
  
# agora, rodar novamente a média  
mean(df$idade, na.rm = TRUE) # na coluna idade, há um valor de idade não informado (NA)

## [1] 36.2

# o comando precisa informar que os NAs devem ser removidos  
 # por isso o comando "na.rm = TRUE"  
  
# procedimento semelhante para o cálculo do desvio padrão.  
sd(df$idade, na.rm = TRUE)

## [1] 24.90381

# mediana  
median(df$idade,na.rm = TRUE)

## [1] 32

# moda  
count(df,escore)

## escore n  
## 1 13 2  
## 2 15 1  
## 3 16 1  
## 4 17 1  
## 5 18 1

# Gráficos

O pacote ggplot2 é um dos que integram a família tidyverse. Ele foi estruturado segundo a Gramática dos Gráficos, em que os comandos são dados por camadas. As três principais camadas são **dados**, **estética** e **geom**. Para realizar as próximas análises iremos utilizar o banco de dados big\_five.rds, que terá que ser importado para o R. Esse banco contém as respostas de 19.719 participantes que responderam a 50 itens relacionados aos cinco grandes fatores de personalidade (extroversão, neuroticismo, cosncienciosidade, amabilidade e abertura). Se você tem um email institucional da UFPE, [clique aqui para baixar o arquivo](https://drive.google.com/file/d/1JEGTkN3T66ubSSVROPVuao9oF9XAzII3/view?usp=sharing). **Salve-o na mesma pasta do R que estão os arquivos utilizados neste disciplina**. Se não tem o email institucional, o arquvio está disponível em formato .csv [nesta página](https://openpsychometrics.org/_rawdata/).

## Preparação do banco de dados

Para importar o arquivo big\_five.rds para dentro do R usamos a função read\_rds() do pacote readr. Para não ter que instalar e carregar esse pacote vamos chamar a função de dentro do pacote usando o script abaixo.

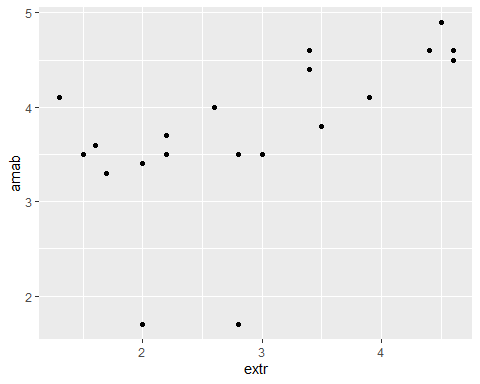
# importar o arquivo big\_five.rds para dentro do R  
  
big\_five <- readr::read\_rds("big\_five.rds")  
  
  
# Seleção de uma pequena parte (20 sujeitos) do banco de dados big\_five, para ficar visualmente mais interessante.  
  
big\_five20 <- big\_five %>% slice(1:20)

## Gráficos de pontos

Num gráfico de pontos, cada ponto representa a interseção entre as variáveis x e y de cada caso. No exemplo a seguir, o gráfico de pontos representa as intersecções entre as variáveis extroversão e amabilidade. Cada ponto representa uma pessoa. De cada ponto é possível projetar perpendiculares aos eixos x e y, correspondentes às pontuações dessa pessoa nas variáveis extroversão e amabilidade, respectivamente.

Então, a tradução do script abaixo ficaria assim: use o pacote ggplot2 (ggplot) e o dataframe big\_five20 (data = big\_five20) para criar um gráfico de pontos (geom\_point), em que os eixos x e y são representados pelos valores dispostos nas variáveis extroversão e amabilidade (aes(x = extr, y = amab)), respectivamente.

ggplot(data = big\_five20) +  
 geom\_point(aes(x = extr, y = amab))

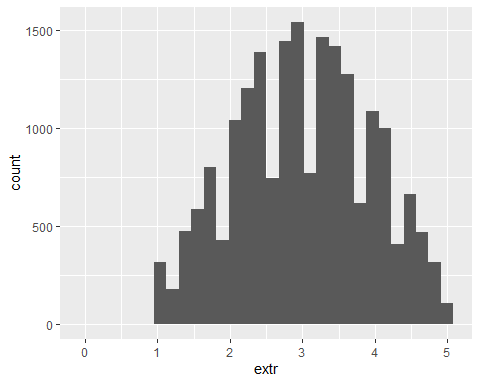


Obs: para uma melhor visualização do gráfico, neste caso, usamos o dataframe big\_five20, que contém apenas os 20 primeiros casos do dataframe big\_five.

## Histograma

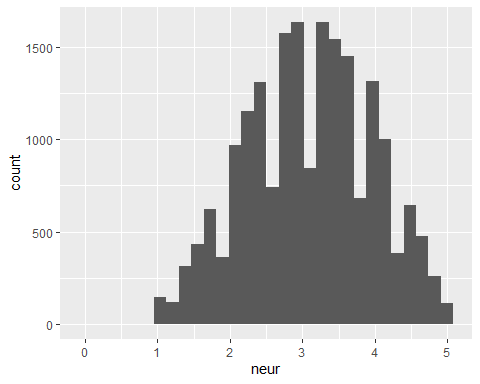
Um histograma representa a distribuição dos valores em uma variável. No eixo x estão os valores que aparecem na variável sob consideração (no caso, fizemos uma gráfico para cada um dos cinco grandes fatores de personalidade); no eixo y está a frequência com que cada valor de x aparece na amostra. Como o eixo y é sempre a frequencia, no caso do histograma (geom\_histogram), precisamos definir apenas a variável que entrará no eixo x.

ggplot(data = big\_five, aes(x = extr)) +  
 geom\_histogram()

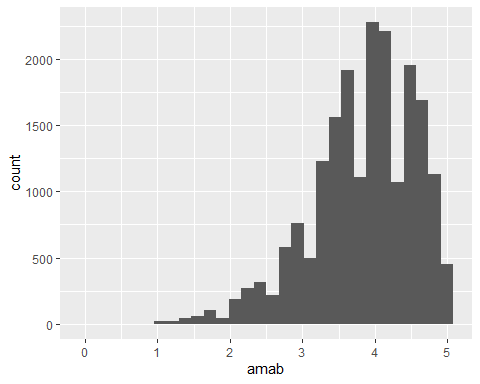


No histograma da variável extroversão, acima, observa-se que há poucos escores baixos (próximos de 1 no eixo x) e altos (próximos de 5 no eixo x) e muitas pessoas com escores ao redor de 3. Note que a distribuição se aproxima de uma distribuição normal. Pode-se aplicar scripts semelhantes para a obtenção de histogramas dos outros quatro fatores.

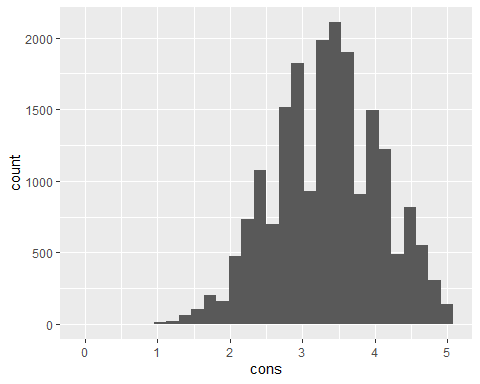
ggplot(data = big\_five, aes(x = neur)) +  
 geom\_histogram()



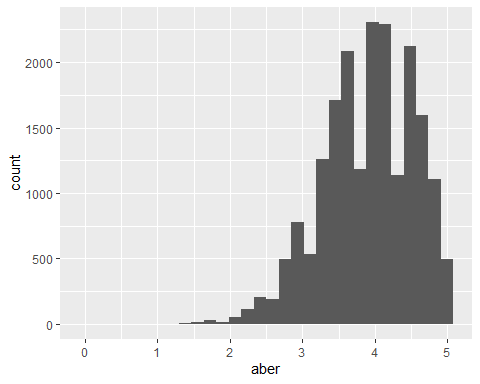
ggplot(data = big\_five, aes(x = amab)) +  
 geom\_histogram()



ggplot(data = big\_five, aes(x = cons)) +  
 geom\_histogram()



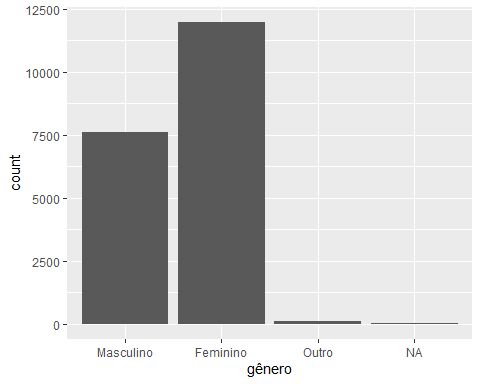
ggplot(data = big\_five, aes(x = aber)) +  
 geom\_histogram()



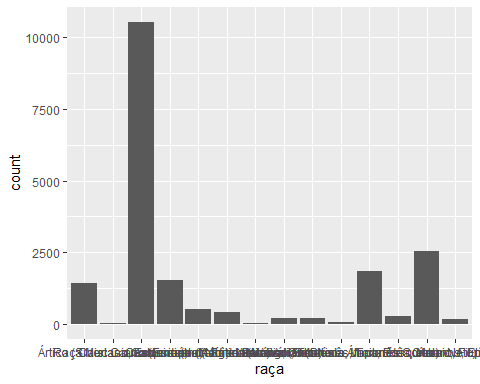
## Gráfico de barras

O gráfico de barras (geom\_bar) é um gráfico com barras retangulares e comprimento proporcional aos valores que ele apresenta. Nos exemplos abaixo, as barras representam a quantidade de pessoas (..count..) por gênero (primeiro gráfico) ou por raça (segundo gráfico)

ggplot(data = big\_five, aes(x = gênero, y = ..count..)) +  
 geom\_bar()



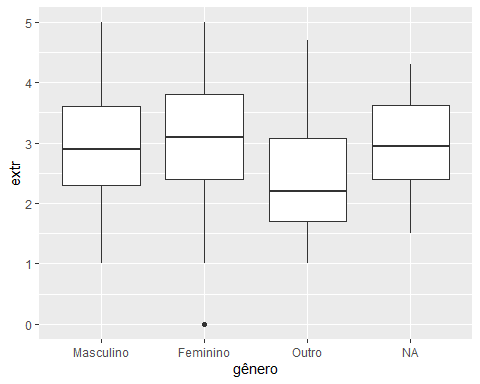
ggplot(data = big\_five, aes(x = raça, y = ..count..)) +  
 geom\_bar()



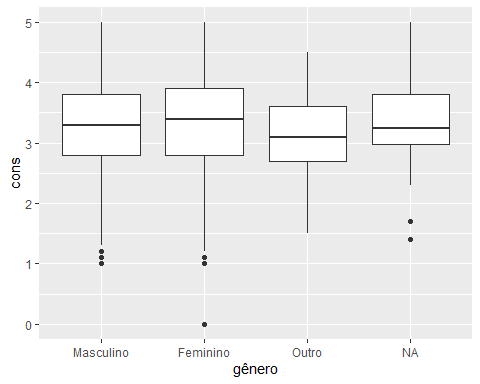
## Boxplot

O boxplot, também chamado de diagrama de caixa é um gráfico que representa a variação de dados por meio de quartis. A haste inferior à caixa representa o primeiro quartil; a caixa representa os quartis 2 e 3, que estão separados pelo traço central correspondente à mediana; e a haste superior à caixa, representa a distribuição dos valores no quarto quartil. Assim, é preciso definir as variáveis que vão em x e y, sendo que y sempre deve ser uma variável contínua.

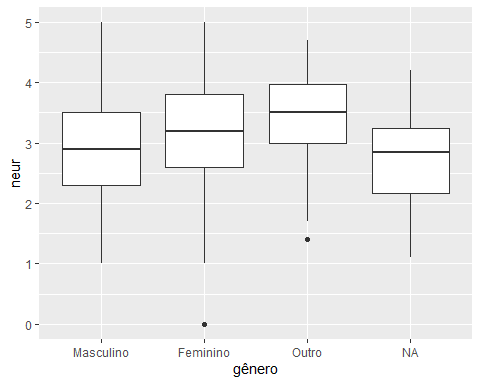
ggplot(data = big\_five, aes(x = gênero, y = extr)) +  
 geom\_boxplot()



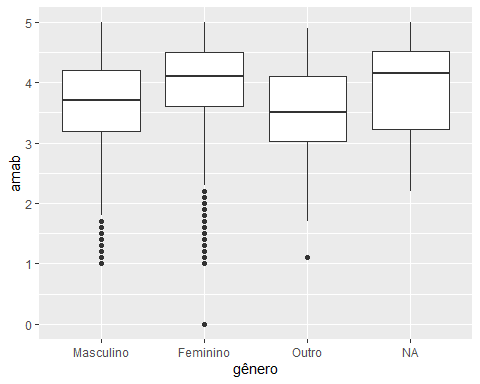
ggplot(data = big\_five, aes(x = gênero, y = cons)) +  
 geom\_boxplot()



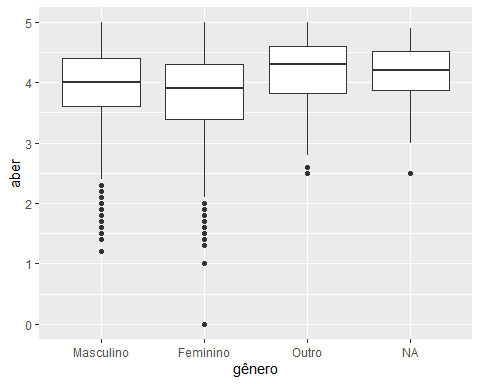
ggplot(data = big\_five, aes(x = gênero, y = neur)) +  
 geom\_boxplot()



ggplot(data = big\_five, aes(x = gênero, y = amab)) +  
 geom\_boxplot()



ggplot(data = big\_five, aes(x = gênero, y = aber)) +  
 geom\_boxplot()



## Gráfico de linhas

Se assemelha a um gráfico de pontos, em que os pontos não estão exatamente dispersos em um plano, mas organizados em uma sequência, o que permite que eles sejam ligados por um segmento de reta, facilitando a visualização das variações entre os pontos. No exemplo abaixo, a linha liga a quantidade de pessoas por gênero.

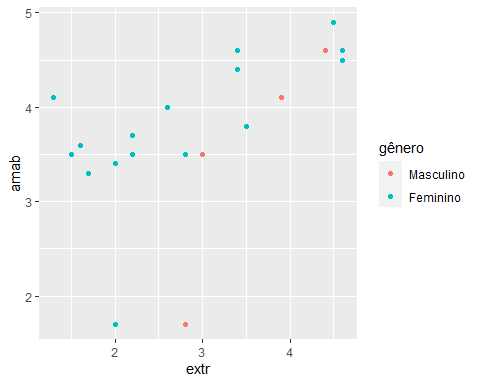
ggplot(data = big\_five) +  
 geom\_line(aes(x = gênero, group = 1), stat = "count")



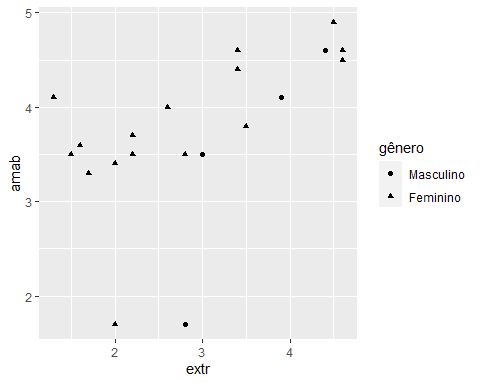
## Modificação de argumentos dentro do geom (color, shape, size)

Uma coisa importante nos gráficos é que eles sejam visualmente atrativos. Para isso, podemos alterar diversos argumentos que os compõem, como cores, formas (shapes) e tamanhos (sizes).

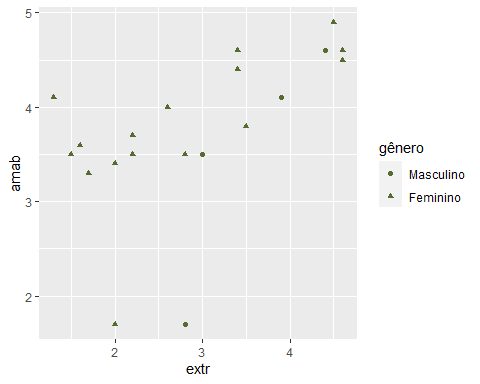
ggplot(data = big\_five20) +  
 geom\_point(aes(x = extr, y = amab, color = gênero))



ggplot(data = big\_five20) +  
 geom\_point(aes(x = extr, y = amab, shape = gênero))

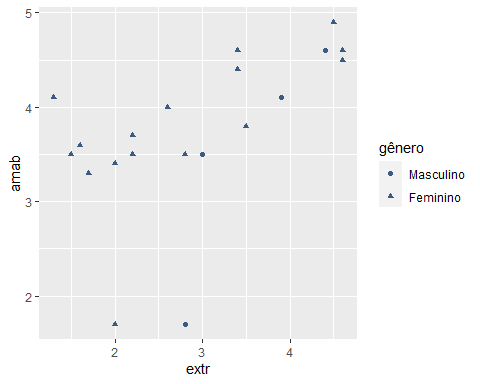


# retirando color de dentro de aes  
ggplot(data = big\_five20) +  
 geom\_point(aes(x = extr, y = amab, shape = gênero),  
 color = "darkolivegreen")



As cores podem ser inseridas pelo seu nome ou pelo seu código. [Clique aqui](http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf) para baixar um arquivo com os nomes das cores do R… e [aqui](https://coolors.co/) para acessar um site gerador de paletas de cores, por código. Os códigos devem ser inseridos após uma hashtag.

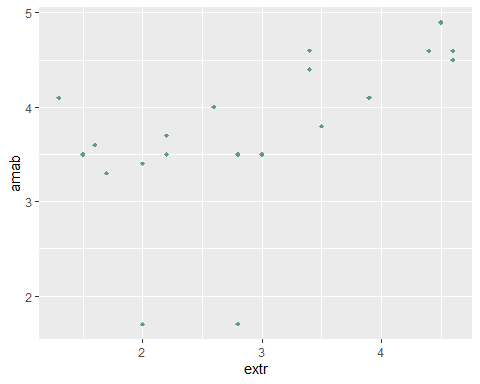
ggplot(data = big\_five20) +  
 geom\_point(aes(x = extr, y = amab, shape = gênero),  
 color = "#3d5a80")



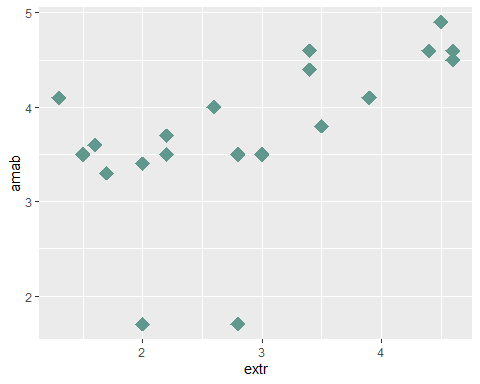
# Site gerador de paletas: https://coolors.co/

Também é possível modificar o formato (shape) dos pontos. [Este site](http://www.sthda.com/english/wiki/ggplot2-point-shapes) tem uma lista dos shapes que podem ser utilizados.

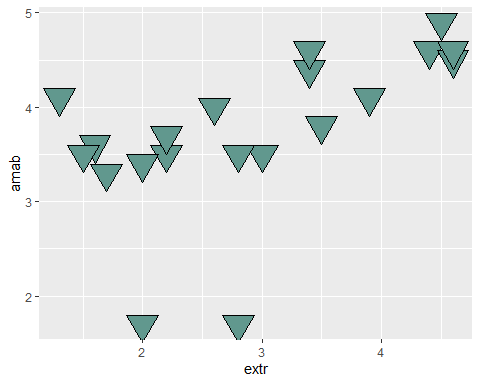
ggplot(data = big\_five20) +  
 geom\_point(aes(x = extr, y = amab),  
 color = "#61988E", shape = 18)



# Shapes possíveis: http://www.sthda.com/english/wiki/ggplot2-point-shapes  
  
ggplot(data = big\_five20) +  
 geom\_point(aes(x = extr, y = amab),  
 color = "#61988E", shape = 18, size = 5)



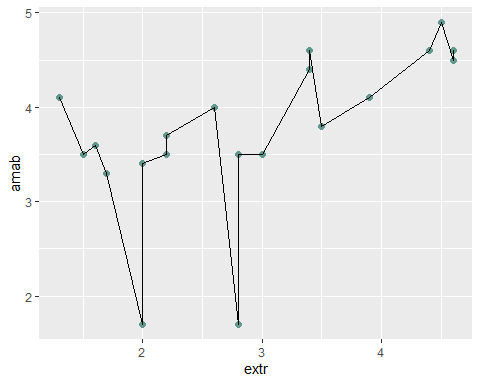
# é possível ajustar a borda (color) e o preenchimento dos pontos (fill)  
# nesse site, os shapes de 21 a 25 permitem ajuste de color e fill  
ggplot(data = big\_five20) +  
 geom\_point(aes(x = extr, y = amab),  
 fill = "#61988E", color = "black", shape = 25, size = 8)



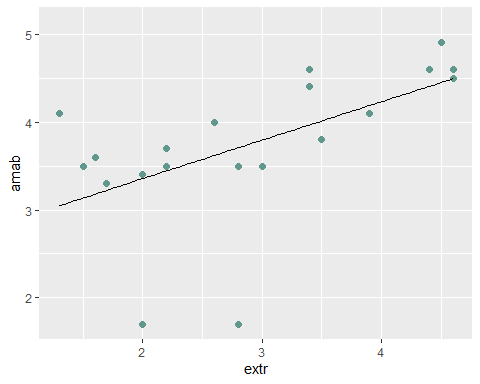
## Adicionando um segundo geom

Como foi dito anteriormente, o ggplot segue o principio de camadas sobrepostas. Portanto, é possível adicionar várias camadas, como outro geom, por exemplo.

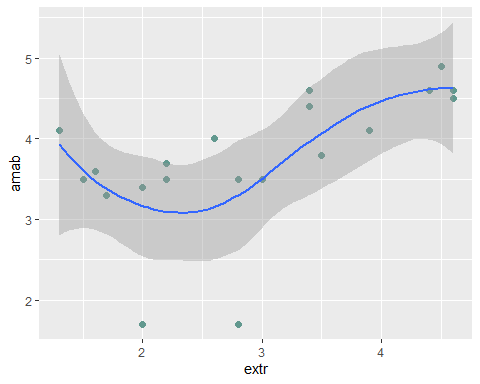
#ggplot(data = big\_five20) +  
# geom\_point(aes(x = extr, y = amab),  
# color = "#61988E", shape = 16, size = 2) +  
# geom\_line(aes(x = extr, y = amab), stat = "smooth",   
# method = "lm")  
  
ggplot(data = big\_five20, aes(x = extr, y = amab)) +  
 geom\_point(color = "#61988E", shape = 16, size = 2) +  
 geom\_line()



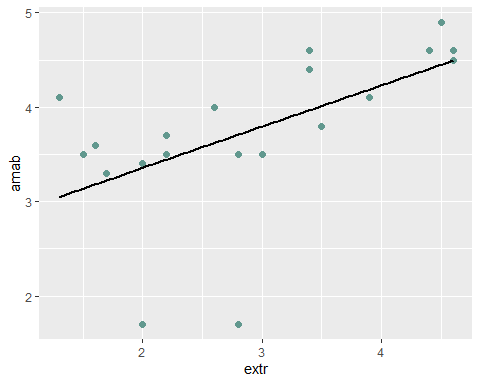
ggplot(data = big\_five20, aes(x = extr, y = amab)) +  
 geom\_point(color = "#61988E", shape = 16, size = 2) +  
 geom\_line(stat = "smooth", method = "lm")



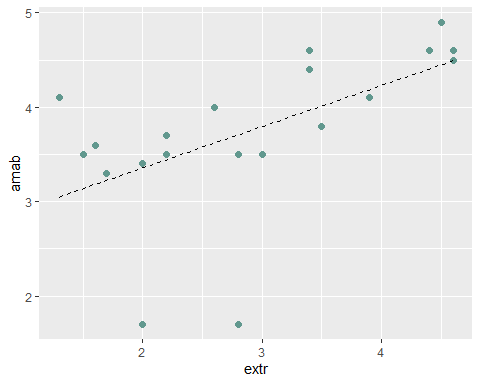
# alterando o tipo de curva de interpolação dos pontos  
  
ggplot(data = big\_five20, aes(x = extr, y = amab)) +  
 geom\_point(color = "#61988E", shape = 16, size = 2) +  
 geom\_smooth()



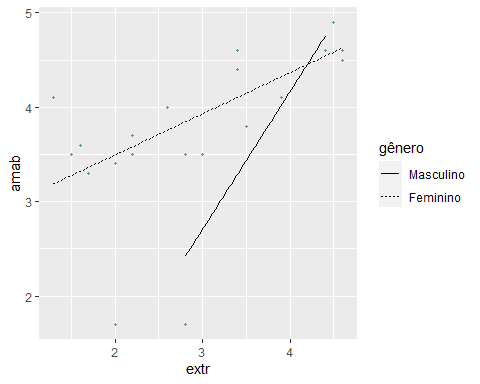
ggplot(data = big\_five20, aes(x = extr, y = amab)) +  
 geom\_point(color = "#61988E", shape = 16, size = 2) +  
 geom\_smooth(method = "lm", se = FALSE, color = "black", size = 1)



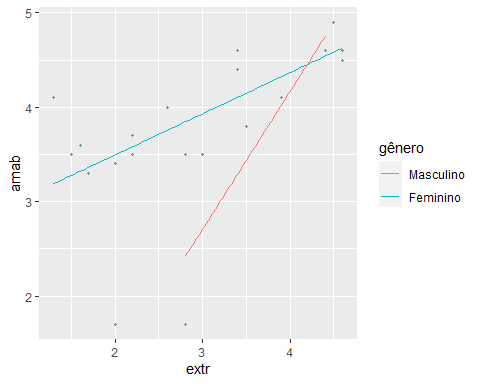
# Mudando o tipo de linha  
ggplot(data = big\_five20, aes(x = extr, y = amab)) +  
 geom\_point(color = "#61988E", shape = 16, size = 2) +  
 geom\_smooth(method = "lm", se = F, color = "black", size = 0.7,  
 linetype = "dashed")



## Especificando aes específicas para um dos geoms  
ggplot(data = big\_five20, aes(x = extr, y = amab)) +  
 geom\_point(color = "#61988E", shape = 16, size = 0.7) +  
 geom\_smooth(method = "lm", se = F, color = "black", size = 0.5,  
 aes(linetype = gênero))



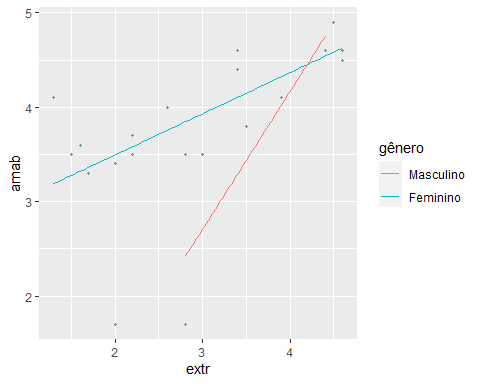
## Especificando aes específicas para um dos geoms  
ggplot(data = big\_five20, aes(x = extr, y = amab)) +  
 geom\_point(color = "#61988E", shape = 16, size = 0.7) +  
 geom\_smooth(method = "lm", se = F, size = 0.5,  
 aes(color = gênero))



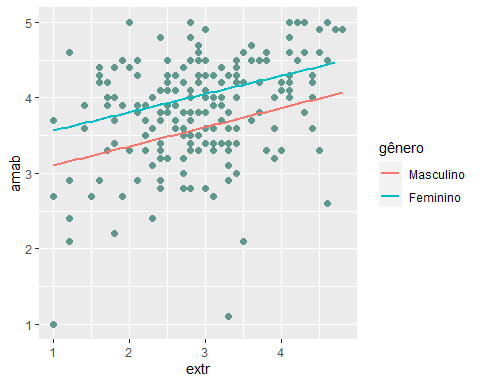
## dplyr e ggplot

Uma opção interessante para a elaboração de gráficos é combinar funções do dplyr com as do ggplot. Por exemplo, ao invés de criar um dataframe com 20 sujeitos, como estávamos fazendo até aqui, podemos selecionar a quantidade de sujeitos desejada por meio da função slice.

# repetindo o último gráfico, mas combinando dplyr com ggplot  
big\_five %>% slice(1:20) %>%   
ggplot(aes(x = extr, y = amab)) +  
 geom\_point(color = "#61988E", shape = 16, size = 0.7) +  
 geom\_smooth(method = "lm", se = F, size = 0.5,  
 aes(color = gênero))

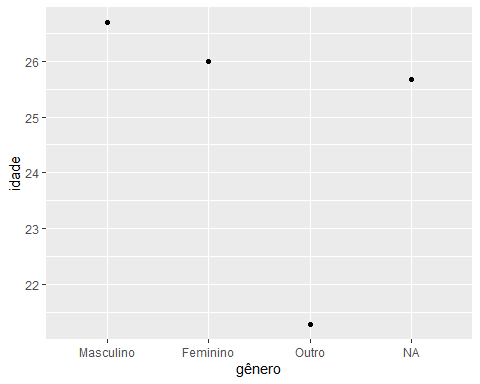


# ou, se quisermos fazer o mesmo gráfico com 1% da amostra  
big\_five %>% slice\_sample(prop = .01) %>%   
 ggplot(aes(x = extr, y = amab)) +  
 geom\_point(color = "#61988E", shape = 16, size = 2) +  
 geom\_smooth(method = "lm", se = F, size = 1,na.rm = TRUE,  
 aes(color = gênero))

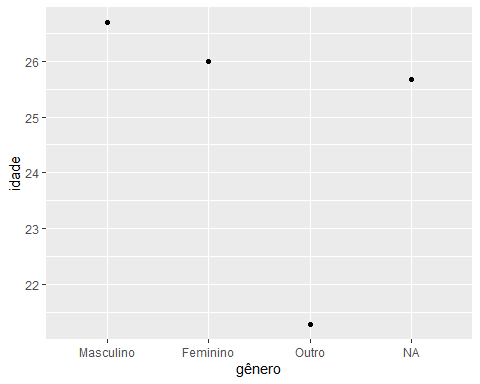


## Usando funções estatísticas

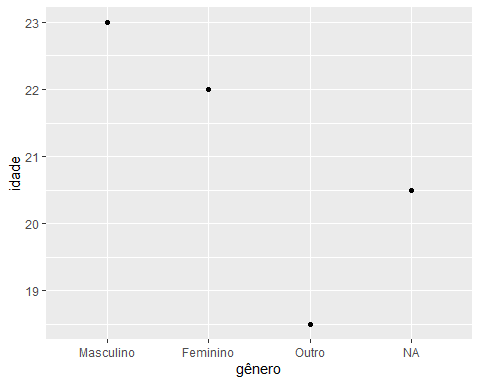
## Usando o geom para representar um "summary"  
### (stat = summary) x stat\_summary()  
  
big\_five %>% filter(idade <= 80) %>% # tem idades erradas no banco  
ggplot() +  
 geom\_point(aes(x = gênero, y = idade), stat = "summary", fun = "mean")



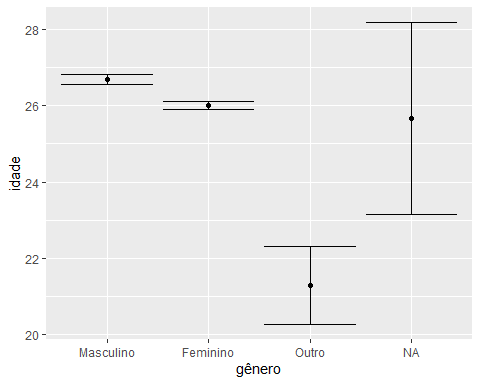
# mesmo gráfico que o anterior, mas usando a função stat\_summary  
big\_five %>% filter(idade <= 80) %>%  
ggplot(aes(x = gênero, y = idade)) +  
 stat\_summary(geom = "point", fun = "mean")



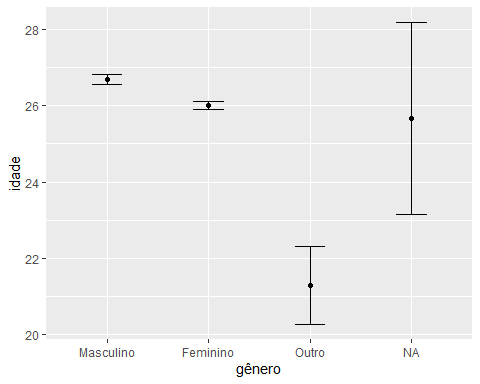
# usando a mediana no lugar da média.  
big\_five %>% filter(idade <= 80) %>%  
ggplot(aes(x = gênero, y = idade)) +  
 stat\_summary(geom = "point", fun = "median")



## Incluindo barras de erros (usando também o summary)  
  
big\_five %>% filter(idade <= 80) %>%  
ggplot(aes(x = gênero, y = idade)) +  
 geom\_point(stat = "summary", fun = "mean") +  
 geom\_errorbar(stat = "summary", fun.data = "mean\_se")



# usando width para melhorar a visualiazação  
  
big\_five %>% filter(idade <= 80) %>%  
ggplot(aes(x = gênero, y = idade)) +  
 geom\_point(stat = "summary", fun = "mean") +  
 geom\_errorbar(stat = "summary", fun.data = "mean\_se", width = 0.3)



# trocando erro da média (mean\_se) pelo mínimo e máximo.  
  
big\_five %>% filter(idade <= 80) %>%  
ggplot(aes(x = gênero, y = idade)) +  
 geom\_point(stat = "summary", fun = "mean") +  
 geom\_errorbar(stat = "summary", fun.min = "min", fun.max = "max")

